Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Επεξεργασία και ανάκτηση πληροφορίας από μουσικές ηχογραφήσεις ή μουσικά αρχεία σε πλαίσιο τεχνολογιών web

---

Φραγκόπουλος Μάρκος-Βασίλειος 2314

Επιβλέπων καθηγητής: Μαλάμος Α.
Επιτροπή αξιολόγησης: Μαλάμος Α. , Παναγιωτάκης Σ. , Παχουλάκης Ι.
Ημερομηνία παρουσίασης:

Σύνοψη

Το Note Recognizer είναι μία διαδικτυακή εφαρμογή συμβατή με όλους τους φυλλομετρητές και τύπους συσκευών, γραμμένη εξ ολοκλήρου σε javascript και HTML5. Η εφαρμογή παρέχει στο χρήστη τη δυνατότητα να δει καταγεγραμμένη τη μουσική πληροφορία που προέρχεται από μία αναπαραγωγή ενός μουσικού αποσπάσματος, και όλα αυτά σε πραγματικό χρόνο. Ο τρόπος λειτουργίας κυρίως βασίζεται σε υπάρχουσες μεθόδους, καθώς και σε νέες που δημιουργήθηκαν και υλοποιήθηκαν για τις απαιτήσεις της εφαρμογής.

Abstract

Note Recognizer is an online application compatible with any major browser and kind of device, coded completely in javascript and HTML5. This application offers to user the ability to see the music information which comes from a reproduction of a music clip recorded in real time. The operation mode based basically from some existing methods, such as new ones that created and implemented for the requirements of this application

Πίνακας περιεχομένων

Πίνακας εικόνων

1 Εισαγωγή

1.1 Περίληψη

Σκοπός της παρούσας εργασίας είναι η δημιουργίας μίας web εφαρμογής που θα απεικονίζει σε πραγματικό χρόνο μία μουσική σύνθεση εισάγεται μέσω μικροφώνου ή αρχείου, χρησιμοποιώντας κοινές μουσικές έννοιες.

Για να υλοποιηθούν τα παραπάνω μελετήθηκαν και κατανοήθηκαν βασικές έννοιες ψηφιακής επεξεργασίας σημάτων, μουσική θεωρία και ερευνήθηκαν οι περισσότεροι αλγόριθμοι αναγνώρισης pitch και onset.

Σε τεχνολογικό επίπεδο σαν βασική γλώσσα προγραμματισμού χρησιμοποιήθηκε η javascript, μελετήθηκε διεξοδικά και χρησιμοποιήθηκε το web audio api καθώς και οι δυνατότητες που παρέχει η γλώσσα σήμανσης html5 και ιδιαίτερα ο canvas. Επίσης χρησιμοποιήθηκαν css και jquery για τη μορφοποίηση του UI. Τέλος χρησιμοποιήθηκε η πλατφόρμα node js για την χρήση εξωτερικών βιβλιοθηκών.

Πρακτικά δοκιμάστηκαν web εφαρμογές που υλοποιούν αλγορίθμους αναγνώρισης pitch, που χρησιμοποιούν το web audio api και που υπολογίζουν το tempo ενός μουσικού κομματιού.

Ακόμα δοκιμάστηκαν μερικές desktop εφαρμογές που υλοποιούν το ίδιο πλαίσιο στόχων όπως η παρούσα εργασία.

Στην εφαρμογή ο χρήστης μπορεί να δει απεικονισμένο σε midi format ή παρτιτούρα ένα μουσικό κομμάτι το οποίο αναπαρήγαγε στο μικρόφωνο ή εισήγαγε σε μορφή αρχείου.

Ως δυνατά σημεία στην εμπειρία του χρήστη μπορούμε να ορίσουμε την απεικόνιση αποτελεσμάτων σε πραγματικό χρόνο και την εύκολη προσβασιμότητα του συστήματος μίας και είναι μία web εφαρμογή.

1.2 Κίνητρο για την διεξαγωγή της εργασίας

Οι ανάγκες ενός σύγχρονου μουσικού σίγουρα δεν περιορίζονται μόνο στη μουσική παιδεία και την έμπνευση, εξέχουσα σημασία έχουν τα εργαλεία που μπορούν να χρησιμοποιηθούν για βελτίωση του αποτελέσματος, εκμάθηση, ανάλυση, διευκόλυνση κτλ.

Η ιδέα γι αυτή την εφαρμογή προέκυψε από τη διαχρονική ενασχόληση του σπουδαστή με τη μουσική σε όλες της τις πτυχές (ηλεκτρονική /οργανική). Κύριο κίνητρο για τη δημιουργία αυτού του εργαλείου είναι η ανάγκη οποιουδήποτε μουσικού να καταγράψει εκείνες τις σκόρπιες στιγμές έμπνευσης και δημιουργίας ώστε να τις διατηρεί ο ίδιος καταγεγραμμένες σαν ιδέες. Για να το κάνει αυτό κάποιος με συμβατικό τρόπο θα πρέπει να έχει βαθύτερες γνώσεις μουσικής ανάλυσης, πολύ χρόνο, αρκετά καλή μνήμη, μεγάλη εμπειρία στη μουσική κλπ. Η πηγή έμπνευσης για την ιδέα δεν περιορίζεται μόνο στην διατήρηση των μουσικών ιδεών αλλά και στην βαθύτερη κατανόηση της μουσικής και στην εκμάθηση αυτής. Για παράδειγμα είναι πολύ χρήσιμο ένας μουσικός απλά να παίζει ένα μουσικό κομμάτι , ιδιαίτερα κάτι ευρέως άγνωστο ή πρωτότυπο, και μετά να μπορεί να το επικοινωνήσει με κάποιον άλλο οργανοπαίκτη χωρίς καν ο δεύτερος να χρειαστεί να τον ακούει να παίζει ή να τον βλέπει.

Επίσης σοβαρό κίνητρο/πρόκληση της εργασίας αυτής είναι η υλοποίηση ενός τέτοιου συστήματος σε διαδικτυακό περιβάλλον αποκλειστικά με την χρήση της γλώσσας προγραμματισμού javascript.

1.3 Ορισμός του προβλήματος

Ως ανάκτηση μουσικής πληροφορίας ορίζουμε την αυτόματη καταγραφή της εκτέλεσης μίας μουσικής ερμηνείας με σκοπό την συμβολική περιγραφή της. Η αυτοματοποίηση αυτής της διαδικασίας αποτελεί σπουδαία πρόοδο για την μουσική κοινότητα αφού η καταγραφή μιας ηχογράφησης αποτελεί και για τον άνθρωπο μία δύσκολη εργασία που γενικά απαιτεί κάποια σχετική εκπαίδευση. Όσο πιο πλούσια είναι η πολυφωνική πολυπλοκότητα της μουσικής σύνθεσης, τόσο μεγαλύτερη εμπειρία απαιτείται να έχει κάποιος στο μουσικό ύφος, στα υπό εξέταση όργανα και στη μουσική θεωρία Παρόλα αυτά, οι έμπειροι μουσικοί είναι ικανοί για την επίλυση πλούσιων πολυφωνιών με μεγάλη ευελιξία όσον αφορά την ποικιλία των οργάνων και των μουσικών υφών καθιστώντας φανερό και αδιαμφισβήτητο το γεγονός ότι τα αυτόματα συστήματα καταγραφής υστερούν στην επίδοση συγκριτικά με τους ανθρώπους που έχουν μουσικές γνώσεις και κατάρτιση. Το κύριο πλεονέκτημα που έχουμε ως άνθρωποι είναι η μοναδική μας ικανότητα στην ταυτοποίηση προτύπων και η μνήμη μας, που μας επιτρέπουν να προβλέψουμε μελλοντικά γεγονότα. Η χρήση μνήμης στην αυτόματη καταγραφή συνήθως συνεπάγεται ένα τεράστιο υπολογιστικό κόστος. Δεν είναι τόσο δύσκολο να συμπεριληφθεί βραχεία μνήμη, αλλά η διατήρηση μιας μνήμης μακράς διαρκείας που σημαίνει να παραμένουν ενεργές πολλές υποθέσεις για διάφορα πλαίσια, έχει μεγάλο κόστος. Η επίλυση ορισμένων ασαφειών που οι άνθρωποι επιλύουν χρησιμοποιώντας τη μακράς διάρκειας μνήμη τους παραμένει μία πρόκληση.[1] Μέχρι σήμερα έχουν

αναπτυχθεί αρκετές εφαρμογές που υλοποιούν κάποιου είδους ανάκτηση μουσικής πληροφορίας όπως τα κουρδιστήρια, οι μετρητές tempo κλπ. Πολύ πιο περιορισμένη είναι η ανάπτυξη συστημάτων που ανακτούν και συνδυάζουν όλες τις απαραίτητες πληροφορίες ώστε να εκφράσουν όσο το δυνατόν πληρέστερα ένα μουσικό κομμάτι. Επίσης ένα πολύ σημαντικό ζήτημα για τέτοιου είδους εφαρμογές είναι ότι είναι πολύ περιορισμένη η καταγραφή και η απεικόνιση σε πραγματικό χρόνο, ιδιαίτερα στα πλαίσια του διαδικτύου.

1.4 Σκοπός και στόχοι εργασίας

Σκοπός αυτής της εργασίας είναι η ανάπτυξη μίας εφαρμογής που θα συνδυάζει σε πραγματικό χρόνο τις πληροφορίες του τονικού ύψους, της χρονικής διάρκειας και του ρυθμού μίας μουσικής σύνθεσης με σκοπό να την αποτυπώσει. Ο χρήστης θα μπορεί να αναπαράγει το μουσικό δείγμα απευθείας στο μικρόφωνο ή μέσω ενός μουσικού αρχείου. Η παρούσα εφαρμογή προσπαθεί να δώσει βαρύτητα και να λύσει ζητήματα (συγκριτικά με άλλες παρόμοιες εφαρμογές) όπως η εύκολη πρόσβαση σε αυτή και η παράθεση αποτελεσμάτων σε πραγματικό χρόνο. Πολύ σημαντική επίσης κρίνεται η απλότητα προς το χρήστη καθώς και η παροχή αποτελεσμάτων ακριβείας.

Οι επιμέρους στόχοι επιγραμματικά είναι οι εξής:

- **Αναγνώριση τονικότητας ηχητικού αποσπάσματος**
  Εξαγωγή τονικού ύψους και ταυτοποίηση νότας μέσω των κατάλληλων αλγορίθμων μετασχηματισμών του σήματος.

- **Αναγνώριση onsets ηχητικού αποσπάσματος**
  Χρήση κατάλληλων αλγορίθμων προεπεξεργασίας και μεταεπεξεργασίας ψηφιακού σήματος καθώς και αναγνώρισης των onsets.

- **Υπολογισμός tempo ηχητικού αποσπάσματος**
  Υλοποίηση των κατάλληλων πράξεων για να υπολογιστεί το tempo βάση των onset.

- **Χαρακτηρισμός νοτών**
  Συνδυασμός των pitch, onset και tempo για τον χαρακτηρισμό των στιγμιότυπων.

- **Απεικόνιση νοτών**
  Απεικόνιση των αντικειμένων σε μουσικά μέτρα σε MIDI format και παρτιτούρα σε πραγματικό χρόνο.

- **Υλοποίηση τεχνικών βελτίωσης αποτελέσματος των αλγορίθμων που θα χρησιμοποιηθούν**
  Χρήση τεχνικών που θα διασφαλίζουν την ορθότητα του αποτελέσματος και θα διέπονται από κάποιους προκαθορισμένους κανόνες

- **Εύκολη πρόσβαση και διαλειτουργικότητα**
  Υλοποίηση της εφαρμογής με τεχνολογίες web ώστε ο χρήστης να μπορεί μέσα από τον browser να την χρησιμοποιήσει και επίσης συμβατότητα με όλους τους τύπους συσκευών (κινητά, tablets κλπ)

- **Απλό περιβάλλον χρήστη**
  Απλοποιημένο περιβάλλον για τον χρήστη χωρίς περιττές πληροφορίες και ρυθμίσεις που έχουν να κάνουν με τεχνικές λεπτομέρειες του συστήματος

1.5 Μέθοδος ανάλυσης & ανάπτυξης πτυχιακής

- έρευνα των απαιτήσεων
- αναζήτηση θεωρητικών διατυπώσεων παρόμοιων συστημάτων
- αναζήτηση τεχνικών που χρειάζονται για την υλοποίηση του συστήματος
- δοκιμές παρόμοιων συστημάτων
- δοκιμές συστημάτων που χρησιμοποιούν ή αναπτύσσουν ένα από τα σημεία που θα υλοποιήσει η παρούσα εργασία.
- Συμπεράσματα για το τι χρειάζεται τεχνικά για να υλοποιηθούν οι θεωρητικές προσεγγίσεις επί του θέματος.
- Κατανόηση τεχνολογικών περιορισμών
- Ορισμός ορίων παρούσας εργασίας
- Λογικός διαχωρισμός του συστήματος σε υποσυστήματα για ευκολότερη κατανόηση και ανάπτυξη
- Έρευνα σχετικά με τις τεχνολογίες που θα χρησιμοποιηθούν σε τεχνικό επίπεδο και επίπεδο γνώσεων
- Αναζήτηση έτοιμων κομματιών κώδικα που υλοποιεί ένα υποσύστημα ή μέρος αυτού και βιβλιοθηκών που θα χρειαστούν

- Δοκιμές των ανωτέρω και διαλογή βάση ποιότητας και σχετικότητας αποτελέσματος
- Συγγραφή κώδικα που υλοποιεί τη βασική δομή της εφαρμογής
- Συναρμογή των εξωτερικών κομματιών κώδικα, των βιβλιοθηκών, του πρωτότυπου κώδικα για την υλοποίηση των υποσυστημάτων
- Δοκιμές σε πραγματικές συνθήκες παράλληλα με τον προγραμματισμό , συνεχής εξαγωγή συμπερασμάτων για τεχνικά όρια, ακρίβειας συστήματος, επόμενα βήματα.
- Διαχωρισμός των επιπέδων ετοιμότητας της εφαρμογής με νέες εκδόσεις
- Υλοποίηση τελικής έκδοσης βάση όλων των παραπάνω
- Δοκιμή τελικής έκδοσης

1.6 Δομή εργασίας

Η παρούσα εργασία προσπαθεί να καλύψει σχεδόν όλα τα σημεία εκείνα που είναι απαραίτητα για την κατανόηση πρώτα του είδους την εφαρμογής και έπειτα της ίδιας. Ξεκινώντας στο κεφάλαιο 2 (*Υπόβαθρο*) γίνεται μία επιγραμματική ανάλυση βασικών εννοιών σε επιστημονικό και μουσικό επίπεδο με στόχο την σφαιρική κατανόηση όρων και μεθόδων που θα χρησιμοποιηθούν αργότερα. Ακολούθως, στο κεφάλαιο 3 (*State of the art*) παρατίθενται περιγραφές ενός μεγάλου μέρους των αλγορίθμων, των μοντέλων και των μεθόδων που έχουν να κάνουν με τις λειτουργίες της εφαρμογής καθώς και μία σειρά από εφαρμογές που καλύπτουν κάποιες ή όλες από τις λειτουργίες που θα καλύψει και η παρούσα εφαρμογή.
Στη συνέχεια, το κεφάλαιο 4 (*Note Recognizer*) περιλαμβάνει μία ενδελεχή παρουσίαση της παρούσας εφαρμογής. Παραθέτει τις προδιαγραφές που πρέπει να ακολουθεί και παρουσιάζει τα μοντέλα, τους αλγόριθμους και τιε μεθόδους που χρησιμοποιήθηκαν ή δημιουργήθηκαν. Αναλύει το σχεδιασμό της εφαρμογής και παρουσιάζει ένα εγχειρίδιο για απλούς χρήστες και ένα για προγραμματιστές. Τέλος, στο κεφάλαιο 5 (*Αποτελέσματα*) παρατίθενται οι δοκιμές που έγιναν στην εφαρμογή και σε κομμάτια της, γενικά συμπεράσματα καθώς και κατευθύνσεις για μελλοντική εργασία και επέκταση.

2 Υπόβαθρο

2.1 Επιστημονικό υπόβαθρο

Ένα θεμελιώδες ζήτημα για να προσεγγιστεί το υπό ανάπτυξη θέμα και να μπορέσει να υλοποιηθεί ένα λογισμικό που θα επιδιώκει την αυτοματοποιημένη ανάκτηση μουσικής πληροφορίας είναι η ανάλυση και κατανόηση του ίδιου του ήχου.

2.1.1 Ήχος

Από φυσική άποψη ένας ήχος παράγεται από τις μεταβολές της πίεσης του αέρα. Παρόλο που η ακριβής διαδικασία διαφέρει κατά περίπτωση οι βασικές αρχές της μετάδοσης και δημιουργίας του ήχου παραμένουν ίδιες. Πιο συγκεκριμένα, τα μόρια όλων των φυσικών σωμάτων διατηρούν σταθερές αποστάσεις από όλα τα γειτονικά τους μόρια. Κατά συνέπεια όταν για οποιονδήποτε λόγο τα μόρια μίας περιοχής πιεστούν και επομένως οι αποστάσεις μεταξύ τους μικρύνουν, τα μόρια της συγκεκριμένης περιοχής επιδιώκουν να επανέλθουν στην αρχική κατάσταση τους συμπιέζοντας τα μόρια των γειτονικών τους περιοχών. Με αυτόν τον τρόπο δημιουργούνται μεταβολές πίεσης που μεταδίδονται με μία ορισμένη ταχύτητα. Αυτές οι μεταβολές αποτελούν ένα ηχητικό κύμα.

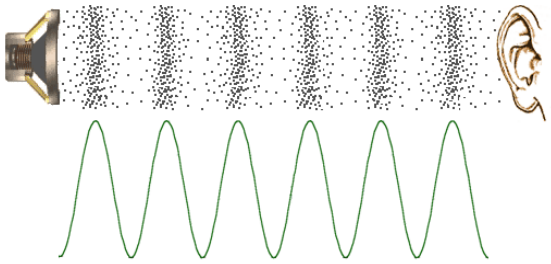


Figure 2.1 Παραλλαγές στην πίεση του αέρα και η αντίστοιχη κυματομορφή [2]

Συνοψίζοντας, θεωρούμε ότι σώματα σε ταλάντωση και μεταβολές πίεσης που μεταδίδονται με μία ορισμένη ταχύτητα αποτελούν την προέλευση των ήχων. Για

παράδειγμα αν χτυπήσουμε τη χορδή μίας κιθάρας τότε η χορδή θα αρχίσει να ταλαντώνεται. Κάθε φορά που η χορδή μετακινείται προς τα πάνω συμπιέζει τα μόρια του αέρα που βρίσκονται πάνω της και τα παραγόμενα ηχητικά κύματα μεταδίδονται μέσω του αέρα μέχρι να συναντήσουν το τύμπανο του αυτιού μας. Οι αρχές λειτουργίας της χορδής είναι κοινές και για άλλες πηγές ήχων. Για παράδειγμα όταν ένας σαξοφωνίστας φυσάει στο σαξόφωνο του, θέτει σε παλινδρομική κίνηση μια στήλη αέρα μέσα στο όργανο του. Ο σαξοφωνίστας μπορεί να μεταβάλει την παλινδρόμηση του αέρα ανοίγοντας και κλείνοντας τρύπες που βρίσκονται κατά μήκος του σαξοφώνου. Η ταλάντωση που εκτελούν τα μόρια ενός σώματος κατά την παραγωγή ενός ήχου μπορεί να είναι αρκετά πολύπλοκη. Για παράδειγμα, η ανθρώπινη παράγεται από ένα συνδυασμό ταλαντώσεων των φωνητικών χορδών οι οποίες προκαλούν την ταλάντωση του αέρα που βρίσκεται στους πνεύμονες, στο λαιμό στο στόμα και στα ιγμόρεια. Ο ήχος μίας κιθάρας ή ενός βιολιού προέρχεται από την ταλάντωση μίας χορδής του οργάνου που θέτει σε παλινδρομική κίνηση τον αέρα ο οποίος βρίσκεται στο αντηχείο του οργάνου. Η χροιά του ήχου που παράγουν τα περισσότερα μουσικά όργανα εξαρτάται από το υλικό, το σχήμα και τις υπόλοιπες φυσικές ιδιότητες του αντηχείου τους.
Η τεχνολογία σήμερα μας δίνει τη δυνατότητα να μετατρέψουμε ένα ηχητικό σήμα σε αναλογικό ηλεκτρικό σήμα και στη συνέχεια σε ψηφιακό ήχο. Σε ψηφιακή μορφή ένα ηχητικό σήμα παριστάνεται από μία σειρά από νούμερα (τα οποία ονομάζουμε δείγματα) τα οποία αντιστοιχούν στην πίεση του αέρα ή στην ηλεκτρική τάση σε διαδοχικές χρονικές στιγμές.

2.1.2 Συχνότητα

Αν θέλουμε να περιγράψουμε κάποιο ήχο που παράγουν δύο πνευστά μουσικά όργανα σαν την τρομπέτα και την τούμπα θα παρατηρήσουμε ότι αν και τα δύο είναι παρόμοια όργανα, η τρομπέτα παράγει πιο υψηλούς ήχους από την τούμπα. Το ηχητικό ύψος είναι ένα υποκειμενικό χαρακτηριστικό που σχετίζεται με ένα αντικειμενικό χαρακτηριστικό, τη συχνότητα. Η συχνότητα ενός συνημιτονικού ηχητικού σήματος αντιστοιχεί στον αριθμό των κύκλων που εκτελεί η συνάρτηση ανά δευτερόλεπτο ενώ μετριέται σε κύκλους ανά δευτερόλεπτο ή Hertz (Hz). Το φάσμα συχνοτήτων (*frequency spectrum*) ενός ηχητικού σήματος αποτελεί το διάγραμμα του πλάτους που έχει κάθε συχνότητα που περιέχεται στο σήμα μας. [3]

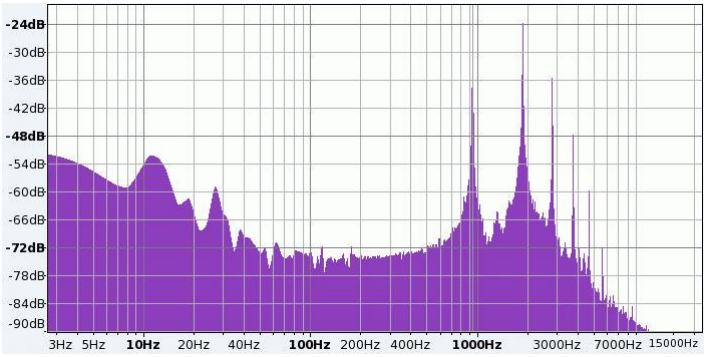


Figure 2.2 Φάσμα συχνοτήτων του ήχου που παράγει μία τρομπέτα

2.2 Μουσικό υπόβαθρο

Η αναγνώριση, η καταγραφή και η απεικόνιση μουσικών δειγμάτων απαιτεί την κατανόηση των χαρακτηριστικά των μουσικών σημάτων και κάποιων βασικών μουσικών εννοιών και δομών.

2.2.1 Χαρακτηριστικά των μουσικών σημάτων

Τα μουσικά σήματα είναι ένα υποσύνολο των ηχητικών σημάτων και έχουν συγκεκριμένα χαρακτηριστικά που μπορούν να ληφθούν υπόψη για την ανάλυσή ενός. Παρατίθενται οι ορισμοί για ενός ποσότητες εκείνες ενός οποίες αντιλαμβάνεται το ανθρώπινο αυτί οι οποίες παίζουν πρωτεύοντα ρόλο στο χαρακτηρισμό ή τον διαχωρισμό των ήχων. Αυτές οι ποσότητες είναι οι λεγόμενες τέσσερις διαστάσεις του ήχου και είναι οι ακόλουθες: η ένταση (*loudness*), η χρονική διάρκεια (*duration*), το τονικό ύψος (*pitch*) και η χροιά (*timbre*).
Η αντίληψη ενός δυναμικής ή ηχηρότητας ενός ακουστικού σήματος δεν συνδέεται με έναν αντίστοιχα απλό τρόπο με ενός φυσικές ιδιότητές του, και τα σχετικά υπολογιστικά μοντέλα για την αντίληψή ενός αποτελούν βασικό πεδίο έρευνας ενός ψυχοακουστικής. Παρόλα αυτά, κατά την επεξεργασία ενός μουσικής είναι βολικό να εκφράζουμε την ηχηρότητα ενός ακουστικού σήματος ως τη μέση τετραγωνική τιμή του (ισχύς) εκφρασμένη σε λογαριθμική κλίμακα (decibel).
Η αντιλαμβανόμενη χρονική διάρκεια ενός ήχου αντιστοιχεί λίγο ως πολύ στη φυσική διάρκειά του σε περιπτώσεις που αυτή μπορεί να προσδιοριστεί σαφώς.
Το pitch είναι μια ιδιότητα εύκολα αντιληπτή από τον άνθρωπο που επιτρέπει την κατηγοριοποίηση των ήχων σε μια κλίμακα συχνοτήτων μεταξύ χαμηλών και υψηλών

τιμών. Ακριβέστερα, το pitch ορίζεται ως η συχνότητα μιας ημιτονοειδούς κυματομορφής που ταιριάζει με τον ήχο που αντιλαμβανόμαστε. Το pitch είναι η συχνότητα του ήχου, ενός γίνεται αντιληπτή από τον άνθρωπο. Η συχνότητα και το pitch δεν ταυτίζονται αλλά συσχετίζονται με μη γραμμικό τρόπο. Η αντίληψη του pitch είναι υποκειμενική και εξαρτάται τόσο από τη συχνότητα όσο και από το επίπεδο ενός πίεσης του ήχου. Η θεμελιώδης συχνότητα (*F0*) είναι η αντιστοιχούσα φυσική σημασία του όρου και ορίζεται μόνο για περιοδικά ή σχεδόν περιοδικά σήματα. Για αυτή την κατηγορία σημάτων, η θεμελιώδης συχνότητα ορίζεται ως το αντίστροφο ενός περιόδου και είναι στενά συνδεδεμένη με το pitch. Σε διφορούμενες καταστάσεις, η περίοδος που αντιστοιχεί στην αντιλαμβανόμενη συχνότητα είναι αυτή που επιλέγεται. Το pitch ενός σύνθετου τόνου μπορεί να γίνει αντιληπτό ακόμα κι αν λείπει η συχνοτική συνιστώσα που αντιστοιχεί στο F0 (απούσα θεμελιώδης). Τα περισσότερα μουσικά όργανα που χρησιμοποιούνται στη δυτική μουσική παράγουν αρμονικούς ήχους καθώς βασίζονται σε έναν αρμονικό ταλαντωτή ενός είναι μια χορδή ή μια στήλη από αέρα. Το φάσμα αυτών των ήχων εμφανίζει μια σειρά από ενός αρμονικές, σε τακτά διαστήματα. Σε έναν ιδανικά αρμονικό ήχο, οι αρμονικές είναι ακέραια πολλαπλάσια ενός θεμελιώδους συχνότητας. Επομένως η F0 ενός αρμονικού ήχου μπορεί να οριστεί ως ο μέγιστος κοινός διαιρέτης των αρμονικών συχνοτήτων.

Το ηχόχρωμα ή χροιά μιας νότας είναι στενά συνδεδεμένο με την πηγή από την οποία έχει παραχθεί ο εκάστοτε ήχος, χαρακτηριστικό γνώρισμα το οποίο μελετάται για το λόγο αυτό σε εργασίες κατηγοριοποίησης των μουσικών οργάνων. Δύο ήχοι που αναπαράγονται από διαφορετικά όργανα, αν και μπορεί να είναι πανομοιότυποι όσον αφορά ενός τιμές pitch και δυναμικής, παραμένουν εύκολα διαχωρίσιμοι λόγω του διαφορετικού ενός ηχοχρώματος. Ο λόγος δεν μπορεί να στηριχθεί σε κάποια ξεχωριστή ιδιότητα του ακουστικού σήματος αλλά εξαρτάται κυρίως από το πόσο 'τραχύ' είναι το ενεργειακό φάσμα του σήματος στο χρόνο και τη χρονική του συνάρτηση. Καθορίζεται ουσιαστικά από τη συμμετοχή των αρμονικών όρων πέραn ενός θεμελιώδους ή βασικής συχνότητας του ήχου και από τα χαρακτηριστικά κορυφώματα ενός περιβάλλουσας του φάσματος. Ενώ λοιπόν το pitch και η δυναμική ενός σήματος μπορούν με απλό τρόπο να αναπαρασταθούν σε κλίμακα δύο διαστάσεων, το ηχόχρωμα αποτελεί ουσιαστικά πολυδιάστατη παράμετρο, και αναπαρίσταται τυπικά από ένα χαρακτηριστικό διάνυσμα.

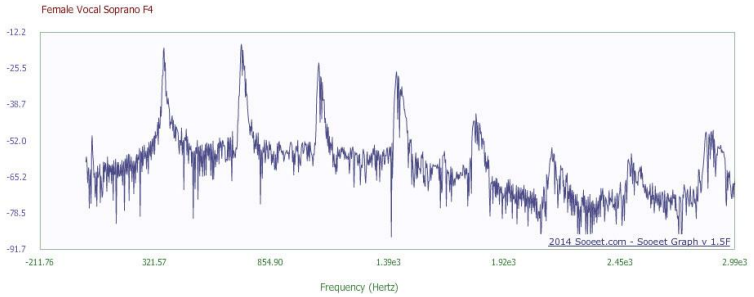


Figure 2.3 Θεμελιώδης συχνότητα και αρμονικές

2.2.2 Τονική δομή

Στη σύγχρονη δυτική μουσική το φάσμα των συχνοτήτων χωρίζεται σε οκτάβες. Μία μουσική νότα μπορεί να αναπαρασταθεί χρησιμοποιώντας ένα γράμμα και έναν αριθμό οκτάβας. Για παράδειγμα, το C3 αναφέρεται στη νότα C από την τρίτη οκτάβα. Οι νότες που διαφέρουν κατά μία οκτάβα έχουν το ίδιο όνομα, δίνουν τη αίσθηση του ίδιου τόνου με μια μεγαλύτερη ή μικρότερη οξύτητα, και οι συχνότητές τους είναι η μία διπλάσια της άλλης. Υπάρχουν διαφορετικοί τρόποι για τη διευθέτηση ενός αριθμού από μουσικές νότες μέσα σε μια οκτάβα και την ανάθεση μίας συχνότητας σε κάθε μία από αυτές. Στη δυτική μουσική, το πιο συνηθισμένο είναι το ισοσυγκερασμένο σύστημα των 12 τόνων, που διαχωρίζει κάθε οκτάβα σε 12 λογαριθμικά ίσα μέρη ή ημιτόνια, τα: A, A#, B, C, C#, D, D#, E, F, F#, G και G# . Ο λόγος των συχνοτήτων δύο διαδοχικών ημιτονίων είναι σταθερός, και ίσος με 2^1/12, έτσι οι συχνότητες των μουσικών νοτών είναι λογαριθμικά κατανεμημένες στον άξονα των συχνοτήτων.

Note	Hz	Note	Hz	Note	Hz	Note	Hz	Note	Hz	Note	Hz	Note	Hz
C1	32.7	C2	65.4	C3	130.8	C4	261.6	C5	523.3	C6	1046.5	C7	2093.0
C#1	34.6	C#2	69.3	C#3	138.6	C#4	277.2	C#5	554.4	C#6	1108.7	C#7	2217.5
D1	36.7	D2	73.4	D3	146.8	D4	293.7	D5	587.3	D6	1174.7	D7	2349.3
D#1	38.9	D#2	77.8	D#3	155.6	D#4	311.1	D#5	622.3	D#6	1244.5	D#7	2489.0
E1	41.2	E2	82.4	E3	164.8	E4	329.6	E5	659.3	E6	1318.5	E7	2637.0
F1	43.7	F2	87.3	F3	174.6	F4	349.2	F5	698.5	F6	1396.9	F7	2793.8
F#1	46.2	F#2	92.5	F#3	185.0	F#4	370.0	F#5	740.0	F#6	1480.0	F#7	2960.0
G1	49.0	G2	98.0	G3	196.0	G4	392.0	G5	784.0	G6	1568.0	G7	3136.0
G#1	51.9	G#2	103.8	G#3	207.7	G#4	415.3	G#5	830.6	G#6	1661.2	G#7	3322.4
A1	55.0	A2	110.0	A3	220.0	A4	440.0	A5	880.0	A6	1760.0	A7	3520.0
A#1	58.3	A#2	116.5	A#3	233.1	A#4	466.2	A#5	932.3	A#6	1864.7	A#7	3729.3
B1	61.7	B2	123.5	B3	246.9	B4	493.9	B5	987.8	B6	1975.5	B7	3951.1

Figure 2.4 Η αντιστοιχία νότας συχνότητας [4]

2.2.3 Ρυθμός

Ο ρυθμός χαρακτηρίζεται από πρότυπα μουσικών μονάδων που εμφανίζονται σε διαφορετικά ιεραρχικά ρυθμικά επίπεδα. Το μήκος των νοτών εξαρτάται από την διάρκεια για την οποία παίζονται, καθώς και από το μουσικό tempo και το μετρικό οπλισμό. Οι βασικές ρυθμικές μονάδες αποκαλούνται beats και ο ρυθμός επανάληψης αυτών των beats δίνει το tempo. Το tempo είναι αντιστρόφως ανάλογο της περιόδου του beat.

Θεωρώντας μια περίοδο beat Tb εκφρασμένη σε seconds, το tempo μπορεί να υπολογιστεί ως T = 60/Tb. Στην περίπτωση δηλαδή που το tempo ισούται με 60, υπάρχει ένα beat ανά δευτερόλεπτο. Ο μετρικός οπλισμός, που δηλώνεται με ένα κλάσμα στην αρχή ενός κομματιού καθορίζει δύο χαρακτηριστικά της μουσικής. Ο αριθμητής δηλώνει τον αριθμό των μετρικών μονάδων μέσα σε ένα μέτρο (απόσταση μεταξύ δύο διαστολών) δηλαδή τα πρότυπα με βάση τα οποία ομαδοποιούνται τα beats. Για παράδειγμα, ένα κομμάτι στο οποίο τα beats ομαδοποιούνται σε ζεύγη, "ένα,δύο,ένα-δύο,ένα-δύο..." συμβολίζεται ότι έχει δίσημο μέτρο. Ο παρονομαστής δηλώνει ποιά είναι η μετρική μονάδα στην οποία αντιστοιχεί το κάθε beat. Ένας τυπικός μετρικός οπλισμός είναι τα 4-4, που αντιστοιχεί σε 4 beats ανά μέτρο, όπου το κάθε beat είναι μια νότα του ενός τετάρτου. Το μήκος των νοτών σε δευτερόλεπτα είναι σχετικό με το μήκος ενός μέτρου. Ο μετρικός οπλισμός και το tempo καθορίζουν τη διάρκεια τόσο του μέτρου όσο και των μεμονωμένων νοτών. [1]

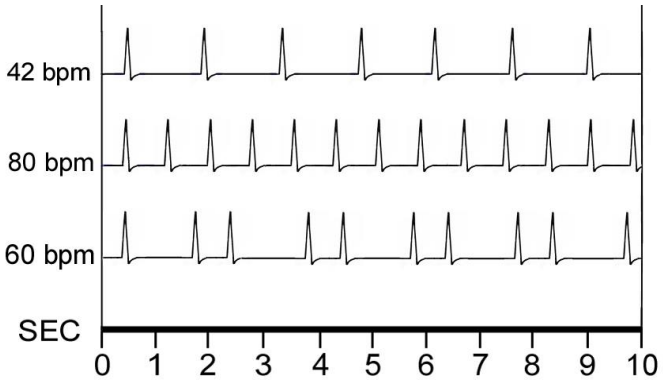


Figure 2.5 Μια αναπαράσταση διαφορετικών BPM ανά μονάδα χρόνου [5]

3 State of the art

3.1 Αλγόριθμοι και μοντέλα γύρω από την ανάκτηση πληροφορίας από μουσικές ηχογραφήσεις ή μουσικά αρχεία σε πλαίσιο τεχνολογιών web

3.1.1 Τεχνολογικό περιβάλλον

3.1.1.1 Μία σύντομη ιστορία του ήχου στο διαδίκτυο

Ο πρώτος τρόπος αναπαραγωγής ήχου στο διαδίκτυο ήταν το <bgsound> tag το οποίο επέτρεπε σε μία ιστοσελίδα να παίζει αυτόματα κάποιο ήχο υποβάθρου όταν κάποιος χρήστης την επισκεπτόταν. Αυτό το χαρακτηριστικό ήταν μόνο διαθέσιμο στο φυλλομετρητή Internet Explorer και δεν τυποποιήθηκε ούτε χρησιμοποιήθηκε ποτέ από κάποιο άλλο φυλλομετρητή. Ο φυλλομετρητης Netscape εφάρμοσε ένα παρόμοιο χαρακτηριστικό μέσω του <embed> tag το οποίο στην ουσία παρείχε ισάξιες δυνατότητες.
Το Flash ήταν ο πρώτος τρόπος αναπαραγωγής ήχου στο διαδίκτυο, συμβατός με όλους τους φυλλομετρητές, όμως παρουσίαζε ένα σημαντικό μειονέκτημα αφού απαιτούσε ένα plugin για να τρέξει. [6]

3.1.1.2 HTML5

Ακλουθώντας τους προκατόχους της HTML 4.01 και XHTML 1.1, η HTML5 είναι μία απάντηση στο γεγονός ότι η HTML και XHTML κατά την κοινή του χρήση στο διαδίκτυο έχουν ένα μείγμα από χαρακτηριστικά διαφόρων προδιαγραφών μαζί με εκείνα που εισήγαγαν τα προϊόντα λογισμικού, όπως φυλλομετρητές κλπ. Περιέχει λεπτομερή μοντέλα επεξεργασίας για να ενθαρρύνει περισσότερο διαλειτουργικές εφαρμογές, επεκτείνει, βελτιώνει και εκλογικεύει τη γλώσσα σήμανσης (markup) και εισαγάγει markup και APIs για πολύπλοκες διαδικτυακές εφαρμογές.
Για τους ίδιους λόγους, η HTML5 είναι ένας πιθανός υποψήφιος για cross-platform εφαρμογές κινητών.

Πολλά χαρακτηριστικά της HTML5 δημιουργήθηκαν για να μπορούν να τρέξουν σε συσκευές με χαμηλές δυνατότητες όπως τα smartphones και τα tablet.Μεταξύ αυτών είναι τα νέα στοιχεία <video>, <audio> και <canvas>. Αυτά τα χαρακτηριστικά σχεδιάστηκαν για να καταστήσουν εύκολη τη συμπερίληψη και το χειρισμό πολυμεσικού και γραφικού περιεχομένου στο διαδίκτυο χωρίς να είναι απαραίτητη η χρήση ιδιόκτητων plugin και API.
Εκτός από το ανανεωμένο markup, η HTML5 καθορίζει κάποια APIs που μπορούν να χρησιμοποιηθούν με τη γλώσσα προγραμματισμού Javascript. Μεταξύ των νέων APIs που υποστηρίζονται είναι τα: Web audio api , Web Storage, Geolocation, Html5 File Api κλπ. [7]

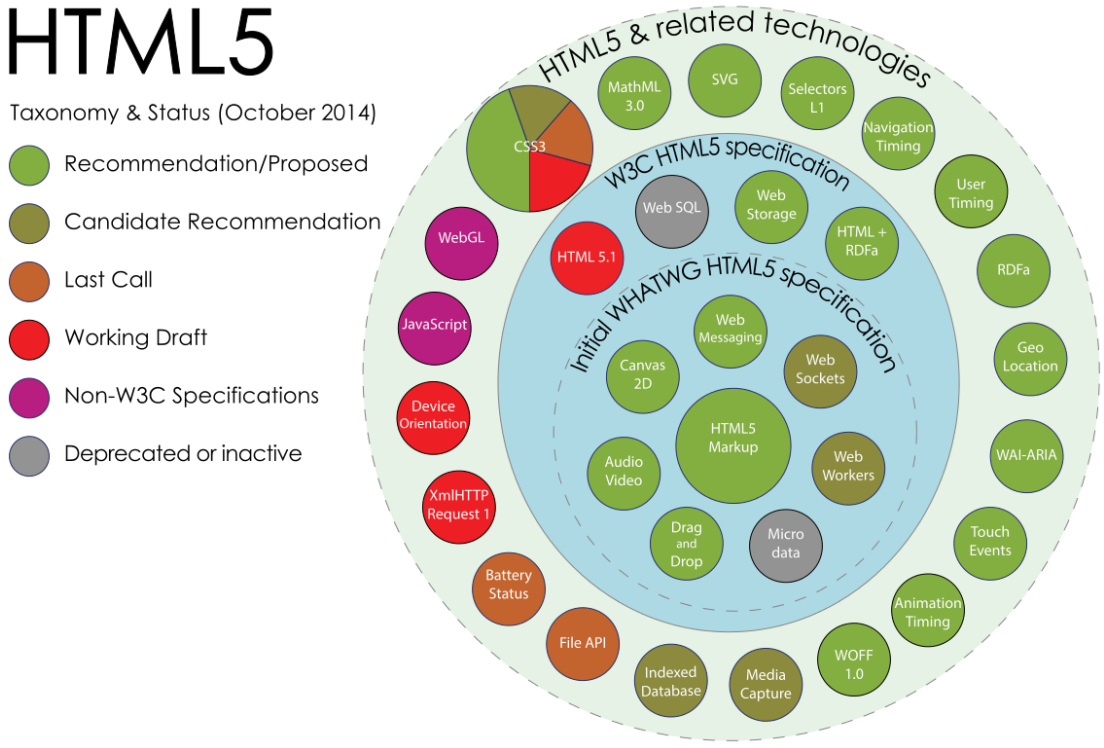


Figure 3.1 API σχετικά με την HTML5 [7]

3.1.1.3 <audio> and Audio Data API

Πιο πρόσφατα οι εταιρίες φυλλομετρητών επικεντρώθηκαν στο <audio> tag της HTML5, το οποίο παρέχει πλήρη υποστήριξη αναπαραγωγής μουσικής σε όλους τους μοντέρνους φυλλομετρητές. Αν και έτσι ο ήχος στο διαδίκτυο δεν απαιτεί πλέον τη χρήση

κάποιου plugin, το <audio> tag παρουσιάζει σημαντικούς περιορισμούς στην υλοποίηση διαδραστικών εφαρμογών. Κάποιοι βασικοί περιορισμοί του στοιχείου <audio>:

- Δεν υπάρχουν ακριβή στοιχεία ελέγχου χρονισμού
- Πολύ χαμηλό όριο για τον αριθμό των ήχων που μπορούν να παίζονται ταυτόχρονα
- Δεν υπάρχει τρόπος για το prebuffer ενός ήχου
- Δεν υπάρχει δυνατότητα να εφαρμοστούν εφέ σε πραγματικό χρόνο
- Δεν υπάρχει τρόπος να αναλυθεί ο ήχος

Υπήρξαν αρκετές προσπάθειες για τη δημιουργία ενός ισχυρού API γύρω από τον ήχο στο διαδίκτυο για να καλυφθούν κάποιοι από τους περιορισμούς που προαναφέρθηκαν. Ένα αξιοσημείωτο παράδειγμα είναι το Audio Data API που σχεδιάστηκε και προτυποποιήθηκε στον Mozilla Firefox. Η προσέγγιση του Mozilla ξεκινάει με ένα στοιχείο <audio> και επεκτείνει το Javascript API του με επιπλέον χαρακτηριστικά. Αυτό το API προσφέρει ένα περιορισμένο γράφημα ήχου και δεν έχει υιοθετηθεί πέραν της πρώτης του εφαρμογής. Αυτή τη στιγμή έχει καταργηθεί από το Firefox για χάρη του Web Audio API. [6]

3.1.1.4 Web Audio Api

Το Web Audio API είναι ένα ολοκαίνουργιο μοντέλο εντελώς διαχωρισμένο από το <audio> tag, αν και υπάρχουν σημεία ενοποίησης με άλλα web APIs.
Ο σκοπός αυτού του API είναι να συμπεριλάβει δυνατότητες που συναντώνται σε σύγχρονες πλατφόρμες παιχνιδιών και σε κάποιες σχετικές με το μιξάρισμα, επεξεργασία και φιλτράρισμα διεργασίες που συναντώνται σε σύγχρονες desktop εφαρμογές επεξεργασίας και παραγωγής ήχου. Το αποτέλεσμα είναι ένα πολύπλευρο API που μπορεί να χρησιμοποιηθεί σε μία πληθώρα διεργασιών σχετικών με τον ήχο. Από παιχνίδια και διαδραστικές εφαρμογές μέχρι προχωρημένες εφαρμογές σύνθεσης, επεξεργασίας και απεικόνισης ήχου. [6]
Σε τεχνικό επίπεδο είναι ένα υψηλού επιπέδου Javascript API για επεξεργασία και σύνθεση ήχου σε διαδικτυακές εφαρμογές. Το κύριο παράδειγμα είναι ενός γραφήματος δρομολόγησης ήχου, όπου ένας αριθμός από AudioNode αντικείμενα είναι μεταξύ τους συνδεδεμένα για να καθορίσουν την συνολική απόδοση του ήχου. Η πραγματική επεξεργασία διεξάγεται κυρίως σε υποκείμενη εφαρμογή (συνήθως σε

βελτιστοποιημένοο Assembly / C / C++ κώδικα), όμως η απευθείας επεξεργασία και σύνθεση με κώδικα Javascript υποστηρίζεται επίσης.

3.1.1.4.1 Βασικές έννοιες πίσω από το Web Audio API

Το Web Audio API περιλαμβάνει το χειρισμό λειτουργιών ήχου μέσα σε ένα audio context, και είναι σχεδιασμένο για σπονδυλωτή δρομολόγηση(*modular routing*). Οι βασικές λειτουργίες ήχου εκτελούνται με audio nodes, τα οποία είναι συνδεδεμένα μεταξύ τους σε ένα γράφημα δρομολόγησης ήχου (*audio routing graph*). Πολλές πηγές ήχου υποστηρίζονται ταυτόχρονα ακόμα και σε ένα audio context. Αυτός ο σπονδυλωτός σχεδιασμός παρέχει την ευελιξία δημιουργίας λειτουργιών ήχου με δυναμικά εφέ. Τα Audio nodes είναι συνδεδεμένα μέσω τον εισόδων και των εξόδων τους, σχηματίζοντας μια αλυσίδα που ξεκινά με μία ή περισσότερες πηγές (*source*), περνώντας από έναν ή περισσότερους κόμβους και καταλήγοντας (προαιρετικά) σε ένα προορισμό(*destination*). Ένα απλό τυπικό workflow για το web audio περιγράφεται παρακάτω:

1. Δημιουργία audio context
2. Μέσα στο audio context δημιουργούμε κάποια πηγή, όπως μικρόφωνο, αρχείο ήχου και ταλαντωτής(*oscillator*)
3. Δημιουργούμε effect nodes όπως αντήχηση(*reverb*), φίλτρο, panner, συμπιεστή(*compressor*)
4. Επιλέγουμε τον τελικό προορισμό (*destination*) του ήχου, για παράδειγμα τα ηχεία του υπολογιστή
5. Συνδέουμε την πηγή με τα εφέ και τα εφέ με τον προορισμό

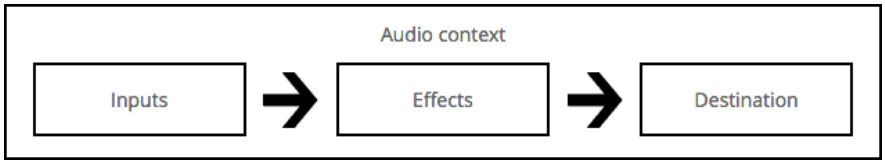


Figure 3.2 Διάγραμμα ροής του Web Audio API [8]

Κάθε είσοδος ή έξοδος αποτελείται από διάφορα κανάλια (*channels*) που αντιπροσωπεύουν μια συγκεκριμένη διάταξη ήχου. Όλες οι διακριτές δομές καναλιών υποστηρίζονται, συμπεριλαμβανομένων των mono, stereo, quad, 5.1, κ.ο.κ.

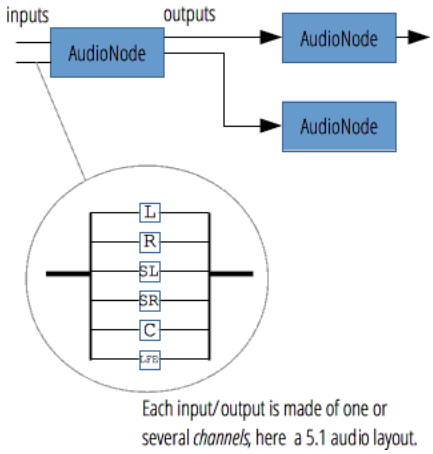


Figure 3.3 Πολλαπλά κανάλια στο WebAudio API [9]

Οι πηγές ήχου μπορούν να προέρχονται από διάφορα σημεία και να παράγονται με πολλούς τρόπους:

- παράγονται απευθείας με κώδικα Javascript από ένα audio node (όπως ένας ταλαντωτής)
- δημιουργούνται από ακατέργαστα PCM δεδομένα (το audio context περιλαμβάνει μεθόδους αποκωδικοποίησης υποστηριζόμενων μορφών ήχου)
- προέρχονται απευθείας από HTML media elements (όπως <audio> και <video>)
- λαμβάνονται από ένα WebRTC MediaStream (όπως από μία κάμερα ή ένα μικρόφωνο) [10]

3.1.1.4.2 Τύποι Web Audio Nodes

Μία από τις κύριες χρήσεις των audio context είναι να δημιουργούν νέα audio nodes. Σε γενικές γραμμές, υπάρχουν διάφορα είδη audio nodes:

Source nodes

Πηγές ήχου όπως audio buffers, live audio inputs, <audio> tags, oscillators και Js processors

Modification nodes

Filters, convolvers, panners, JS processors κλπ

Analysis nodes

Analyzers και JS processors

Destination nodes

Audio outputs και offline processing buffers[6]

3.1.1.4.3 Analyser Node

Η διεπαφή AnalyserNode αντιπροσωπεύει ένα κόμβο που είναι ικανός να παρέχει πληροφορίες στο πεδίο της συχνότητας και του χρόνου σε πραγματικό χρόνο. Είναι ένα audio node το οποίο περνάει το ηχητικό σήμα απαράλλακτο από την είσοδο στην έξοδο, όμως μας επιτρέπει να παίρνουμε τα παραγόμενα δεδομένα και να τα επεξεργαζόμαστε με javascript. Τα παραγόμενα δεδομένα βασίζονται σε FFT ανάλυση (μέγιστο μέγεθος παραθύρου 2048) ενός συγκεκριμένου μεγέθους buffer. Οι τιμές κανονικοποιούνται ώστε να είναι μεταξύ 0 και 1. [11]

3.1.1.4.4 ScriptProcessor Node and Audio Worker Node

Η διεπαφή ScriptProcessorNode επιτρέπει την παραγωγή, επεξεργασία και ανάλυση ενός ηχητικού σήματος χρησιμοποιώντας Javascript.Είναι ένα audio node το οποίο συνδέεται με δύο buffer, η μία περιλαμβάνει τα εισαγόμενα δεδομένα και η άλλη τα τα επεξεργασμένα δεδομένα εξόδου. Ένα συμβάν (*event*) αποστέλλεται στο αντικείμενο κάθε φορά που υπάρχουν νέα δεδομένα στο input και κάθε φορά που γεμίζει το output buffer με δεδομένα ένας χειριστής συμβάντος (*event handler*) τερματίζει τη διαδικασία. Το μέγεθος buffer πρέπει να είναι δύναμη του 2 ανάμεσα στα 256 και 16384. Μικροί αριθμοί ελαχιστοποιούν την καθυστέρηση όμως μεγάλοι αριθμοί είναι απαραίτητοι για την αποφυγή δυσλειτουργιών στον ήχο.
Αυτός ο τύπος κόμβου είναι προς κατάργηση για να αντικατασταθεί από τον AudioWorkerNode. [12]
Η διεπαφή AudioWorkerNode αντιπροσωπεύει ένα κόμβο ο οποίος διαδρά με ένα Worker thread για την απευθείας παραγωγή, επεξεργασία και ανάλυση ήχου. Ο χρήστης δημιουργεί ένα ξεχωριστό σενάριο επεξεργασίας ήχου για τον Worker, που φιλοξενείτε

μέσα στο AudioWorkerGlobalScope και τρέχει μέσα στο audio processing thread, αντί του κεντρικού thread διεπαφής χρήστη. Αν και αυτός ο τύπος κόμβου έχει προτυποποιηθεί από το w3c δεν έχει ακόμα υλοποιηθεί στους δημοφιλείς φυλλομετρητές. [13].

3.1.1.4.5 DynamicsCompressorNode

Η δυναμική συμπίεση χρησιμοποιείτε ευρέως στη μουσική παραγωγή και στον ήχο παιχνιδιών . Χαμηλώνει την ένταση των δυνατότερων σημείων ενός σήματος και αυξάνει την ένταση των χαμηλότερων. Γενικά ένας δυνατότερος και πιο πλούσιος ήχος μπορεί να επιτευχθεί. Είναι πολύ σημαντικό σε παιχνίδια και μουσικές εφαρμογές όπου ένας μεγάλος αριθμός διαφορετικών ήχων ακούγονται ταυτόχρονα να υπάρχει δυνατότητα ελέγχου της συνολικής έντασης του σήματος και βοήθεια στην αποφυγή ψαλιδίσματος (*clipping*) του ήχου. Οι ιδιότητες του αντικειμένου που μπορούν να ρυθμιστούν είναι τα: κατώφλι (*threshold*), knee, σχέση (*ratio*), μείωση (*reduction*), attack, release. [14]

3.1.1.4.6 BiquadFilterNode

Στην επεξεργασία σήματος, ένα φίλτρο είναι μία συσκευή ή διαδικασία η οποία αφαιρεί από το σήμα κάποιο ανεπιθύμητο στοιχείο ή χαρακτηριστικό. Το χαρακτηριστικό στοιχείο των φίλτρων είναι η πλήρης ή μερικής κατάργηση κάποιων πτυχών του σήματος. Συχνά αυτό σημαίνει αφαίρεση κάποιων συχνοτήτων από όλες ώστε να καταστείλουν σήματα παρεμβολής και να μειώσουν τον θόρυβο υποβάθρου. [15]
Η διεπαφή BiquadFilterNode αναπαριστά ένα απλό χαμηλής τάξεως φίλτρο. Είναι ένα AudioNode που μπορεί να αντιπροσωπεύσει διαφορετικά είδη φίλτρων, συσκευές ελέγχου τόνου και γραφικούς εξισωτές (*graphic equalizers*). [16]

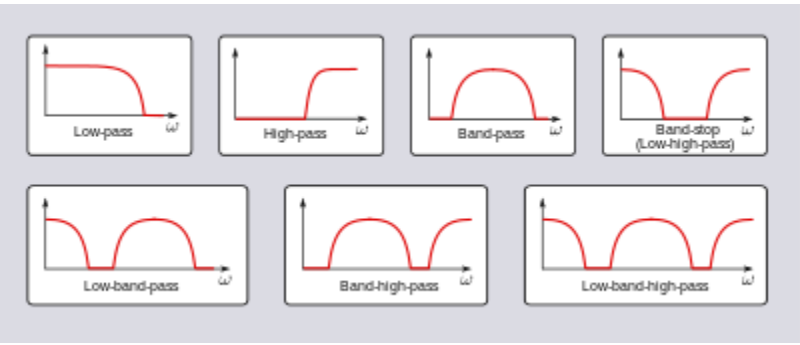


Figure 3.4 Διάφοροι τύποι φίλτρων [17]

3.1.2 Απεικόνιση σημάτων στο πεδίο χρόνου/συχνότητας

Τα ηλεκτρικά σήματα έχουν δύο ειδών αναπαραστάσεις, μία στο πεδίο του χρόνου μία στο πεδίο της συχνότητας. Στο πεδίο της συχνότητας τάση ή το ρεύμα εκφράζεται ως συνάρτηση του χρόνου. Οι περισσότεροι άνθρωποι είναι σχετικά άνετοι με παραστάσεις των σημάτων στο πεδίο του χρόνου. Σήματα μπορεί επίσης να αντιπροσωπεύεται από έκτασης και φάση ως μία συνάρτηση της συχνότητας. Τα σήματα μπορεί επίσης να αντιπροσωπεύονται από ένα πλάτος και μία φάση ως συνάρτηση της συχνότητας.

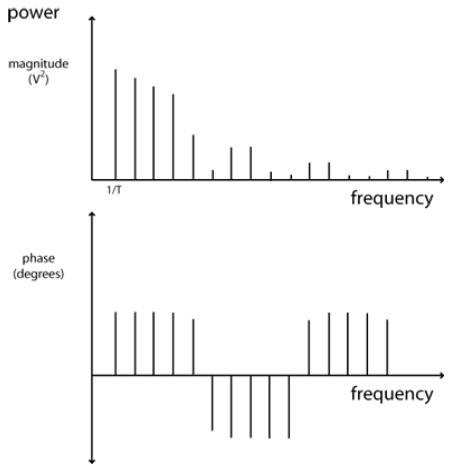


Figure 3.5 Φάσμα ισχύος ενός περιοδικού σήματος [6]

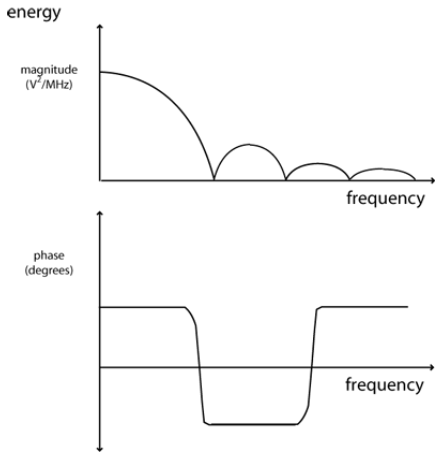


Figure 3.6 Φάσμα ενέργειας ενός σήματος για περιορισμένο χρονικό διάστημα [6]

Παραστάσεις στο πεδίο των συχνοτήτων είναι ιδιαίτερα χρήσιμες κατά την ανάλυση γραμμικών συστημάτων. Οι πηγές σήματος οι παρεμβολές συχνά ορίζονται στο πεδίο του χρόνου. Ωστόσο, η συμπεριφορά του συστήματος και οι μετασχηματισμοί σήματος είναι πιο βολικό και έξυπνο όταν επεξεργάζονται στο πεδίο της συχνότητας.
Η μουσική αποτελείται από πολλούς ήχους που παίζονται ταυτόχρονα. Οι ήχοι που παράγονται από μουσικά όργανα μπορεί να γίνουν πολύ πολύπλοκοι, καθώς ο ήχος ανακλάται μέσω διαφόρων μερών του οργάνου και διαμορφώνεται με μοναδικούς τρόπους. Παρόλα αυτά όλοι αυτοί οι μουσική τόνοι έχουν ένα κοινό, φυσικά, είναι όλοι περιοδικές κυματομορφές.
Η σχέση μεταξύ των γραφημάτων πεδίου χρόνου και πεδίου συχνοτήτων βασίζεται στην ιδέα της αποσύνθεσης fourier (*fourier decomposition*). Τα ηχητικά κύματα είναι συνήθως κυκλικά στη φύση. Μαθηματικά, τα περιοδικά ηχητικά κύματα μπορούν να φανούν ως το άθροισμα των πολλαπλών απλών ημιτονοειδών κυμάτων διαφορετικών συχνοτήτων και πλάτους. Όσο περισσότερα από αυτά τα κύματα ημιτόνου προσθέσουμε μαζί, τόσο καλύτερη προσέγγιση της αρχικής συνάρτησης μπορούμε να πάρουμε. Μπορούμε να πάρουμε ένα σήμα και να βρούμε συνιστώσες ημιτονοειδών κυμάτων με την εφαρμογή ενός μετασχηματισμού Fourier. Πολλοί αλγόριθμοι υπάρχουν για μία τέτοια αποσύνθεση, μεταξύ των οποίων καλύτερος θεωρείτε ο ταχύς μετασχηματισμός Fourier (*FFT*). Το Web Audio API διαθέτει μία υλοποίηση αυτού του αλγορίθμου.
Σε γενικές γραμμές, μπορούμε να πάρουμε ένα ηχητικό κύμα, να συμπεράνουμε τα συστατικά της ανάλυσης του και να σχεδιάσουμε τη συχνότητα και την ένταση σαν σημεία σε ένα καινούργιο γράφημα για να πάρουμε μία απεικόνιση του πεδίου της συχνότητας.

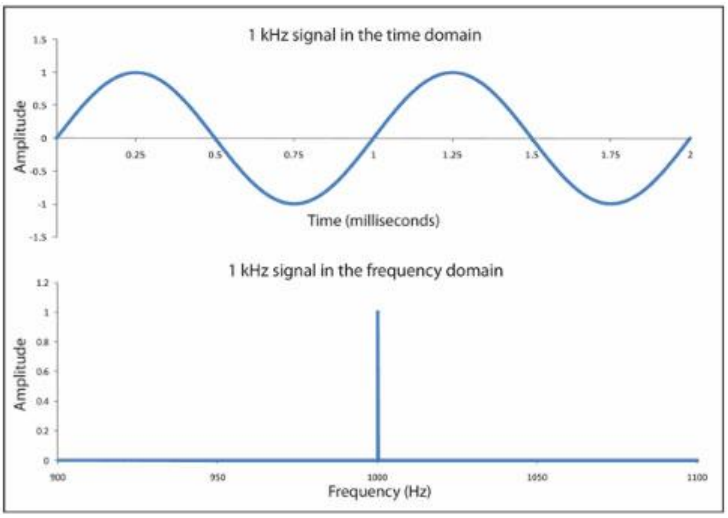


Figure 3.7 Ένα τέλειο ημιτονοειδές ηχητικό κύμα 1 KHz αναπαριστάται και στο πεδίο του χρόνου και στο πεδίο της συχνότητας [6]

Κοιτώντας το πεδίο της συχνότητας μπορεί να μας δώσει μία καλύτερη αίσθηση των ιδιοτήτων του ήχου, συμπεριλαμβανομένου του περιεχομένου τονικού ύψους, την ποσότητα του θορύβου και πολλά ακόμα.
Ήχοι που παράγονται από πραγματικά μουσικά όργανα έχουν αρμονικότητα, έτσι μια A4 παιγμένη σε ένα πιάνο έχει μία αναπαράσταση τελείως διαφορετική από την ίδια νότα παιγμένη από μία τρομπέτα.

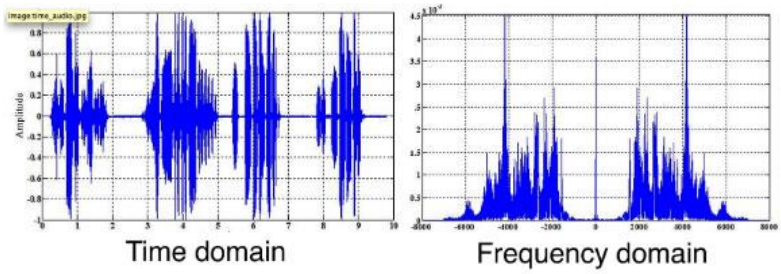


Figure 3.8 Ένα σύνθετο ηχητικό κύμα παρουσιάζεται τόσο στο πεδίο του χρόνου όσο και στης συχνότητας [6]

Η αναπαράσταση στο πεδίο του χρόνου προχωράει από τα αριστερά προς τα δεξιά. Η αναπαράσταση στο πεδίο της συχνότητας είναι η ανάλυση των συχνοτήτων της κυματομορφής σε μια στιγμή, έτσι θα μπορούσε να αλλάζει πιο γρήγορα και λιγότερο προβλέψιμα. [6]

3.1.3 Εκτίμηση της αρχής (onset) της νότας

Ο όρος onset σε ένα μουσικό σήμα περιγράφει την αντιληπτή αρχή ενός διακριτού γεγονότος. Συνεπώς ο όρος ανίχνευση onset αναφέρεται στην ανίχνευση της αρχής ενός διακριτού γεγονότος σε ένα ηχητικό σήμα. Αυτά τα γεγονότα μπορεί να είναι ήχοι με pitch ή χωρίς. Ο στόχος ενός συστήματος ανίχνευσης των onsets είναι να διαχωρίσει το σήμα σε μικρότερες μονάδες (με μονωμένες νότες, συγχορδίες, κτλ.). Η εύρεση των onsets είναι χρήσιμη σε διάφορες εφαρμογές, από την εκτίμηση του tempo και τον εντοπισμό των beats, μέχρι την εκτίμηση των θεμελιωδών συχνοτήτων, εφόσον η αρχή των νοτών μπορεί να χρησιμοποιηθεί για τον τεμαχισμό του σήματος. Η πιο συνηθισμένη προσέγγιση του προβλήματος της εύρεσης των onsets είναι η αναζήτηση κάποιων "μεταβατικών" περιοχών στο σήμα. Μια μεταβατική περιοχή μπορεί να προσδιορίζεται από μία απότομη αύξηση της ενέργειας, μια αλλαγή στο φάσμα βραχέως χρόνου του σήματος, στις στατιστικές ιδιότητές του, κτλ. [18]

Ένα βασικό ζήτημα εδώ είναι να γίνει σαφής διάκριση μεταξύ των συναφών εννοιών των transients, onsets και attacks. Ο λόγος που γίνεται αυτή η διάκριση είναι ότι κάθε εφαρμογή έχει διαφορετικές ανάγκες. Οι ομοιότητες και οι διαφορές μεταξύ αυτών των βασικών εννοιών είναι κάτι που πρέπει να λαμβάνεται υπόψη ενώ εξίσου σημαντικό είναι να κατηγοριοποιηθούν όλες οι διαφορετικές προσεγγίσεις. Στο παρακάτω σχήμα φαίνεται μία απομονωμένη νότα, παρατηρούμε ότι είναι δύσκολο να διαχωρίσουμε αυτές τις έννοιες.

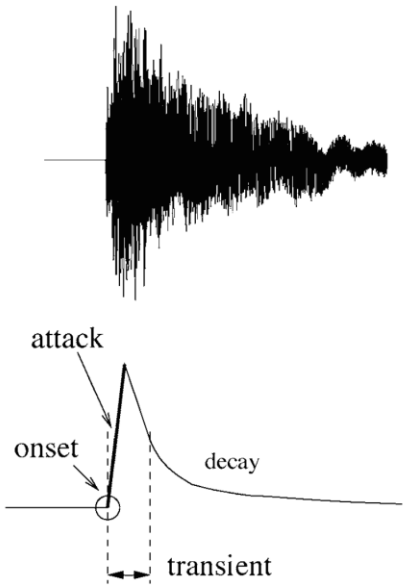


Figure 3.9 Γεγονότα ενός ηχητικού σήματος [19]

- **Attack** είναι ο αρχικός χρόνος που παίρνει στο επίπεδο του ήχου να ανέβει από μηδενικός στην κορυφή, ξεκινώντας όταν παιχτεί ή νότα.
- Ένα γεγονός **transient** (*μεταβατικός χρόνος*) είναι μια βραχύβια έκρηξη της ενέργειας σε ένα σύστημα που προκαλείται από μια ξαφνική αλλαγή φάσης. Η πηγή της ενέργειας του transient μπορεί να είναι ένα εσωτερικό γεγονός ή ένα κοντινό γεγονός. Η ενέργεια στη συνέχεια καταλήγει σε άλλα κομμάτια του συστήματος και τυπικά εμφανίζεται ως μια σύντομη έκρηξη ταλάντωσης.
- Το **onset** μιας νότας είναι η χρονική στιγμή στην οποία ξεκινάει το transient (ή η στιγμή που μπορούμε να διακρίνουμε την αρχή του). Κάποιες εργασίες ορίζουν τα onsets σαν τη χρονική στιγμή που εμφανίζεται το attack. Όλες οι μουσικές νότες έχουν ένα onset αλλά δεν περιλαμβάνουν απαραίτητα ένα αρχικό transient.

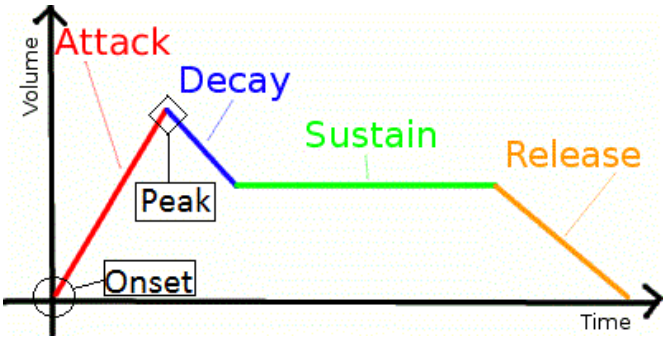


Figure 3.10 Πιό αναλυτικά τα γεγονότα ενός ηχητικού σήματος [20]

Επίσης στο σχήμα διακρίνουμε μαρκαρισμένες κάποιες ακόμα έννοιες:

- **Decay** είναι ο χρόνος ανάμεσα στο attack και το προκαθορισμένο επίπεδο sustain του οργάνου.
- **Sustain** είναι το επίπεδο του ήχου κατά τη διάρκεια της νότας, μέχρι να αφεθεί η νότα.
- **Release** είναι ο χρόνος μέχρι το επίπεδο του ήχου από το τέλος του sustain φτάσει το μηδέν.

Σε ένα πραγματικό σήμα (το οποίο εν γένει είναι πολυφωνικό και πολύ πιθανό να έχει υπερτεθεί και θόρυβος) οι παραπάνω έννοιες δεν μπορούν να διακριθούν με ακρίβεια. Έτσι, τα onsets δεν μπορούν να ανιχνευθούν απευθείας από το σήμα ή την παραγοντοποίηση του στο πεδίο του χρόνου. Πρέπει να βρεθεί ένα ενδιάμεσο σήμα που να αντανακλά, σε μια απλοποιημένη μορφή, την τοπική δομή του αρχικού σήματος. Αυτό το ενδιάμεσο σήμα αποκαλείται συνάρτηση ανίχνευσης (*detection function*).

Οι περισσότεροι υπάρχοντες αλγόριθμοι εύρεσης των onsets ακολουθούν ένα συγκεκριμένο μοτίβο, το οποίο έχει την παρακάτω ακολουθία:

1. Προεπεξεργασία του ανεπεξέργαστου ηχητικού σήματος με στόχο την αύξηση των επιδόσεων των ακόλουθων βημάτων.
2. Δημιουργία μίας συνάρτησης ανίχνευσης, π.χ μία συνάρτηση της οποίας οι κορυφές ταυτόχρονα θα ταυτίζονται με τα onset.
3. Εφαρμογή ενός αλγορίθμου επιλογής κορυφών στη συνάρτηση ανίχνευσης ώστε να επιλεχθούν οι κατάλληλες κορυφές

3.1.3.1 Προ-επεξεργασία

Ο όρος προ-επεξεργασία υποδεικνύει το μετασχηματισμό του αρχικού σήματος με σκοπό την εξασθένιση ή την ενίσχυση κάποιων πτυχών του σήματος, ώστε να απλοποιηθεί ο επιθυμητός στόχος. Η προεπεξεργασία είναι ένα προαιρετικό βήμα. Στην βιβλιογραφία αναφέρονται διάφορες μέθοδοι προ-επεξεργασίας που διευκολύνουν το πρόβλημα της ανίχνευσης των onsets. [1]
Μία μέθοδος προ-επεξεργασίας είναι η υλοποίηση STFT (*short time fourier transform*) στο ψηφιακό σήμα.

Ο μετασχηματισμός STFT περιλαμβάνει τα εξής βήματα:

1. Τεμαχισμός του σήματος σε παράθυρα (*Window Slicing*)
2. Εφαρμογή παραθυριοποίησης Hamming (*Hamming Window*)
3. Κανονικοποίηση μέτρου (*Normalization*)
4. Προσθήκη μηδενικών (*Zero Padding*)
5. Μετασχηματισμός στο πεδίο της συχνότητας (*Fast Fourier Transform*)

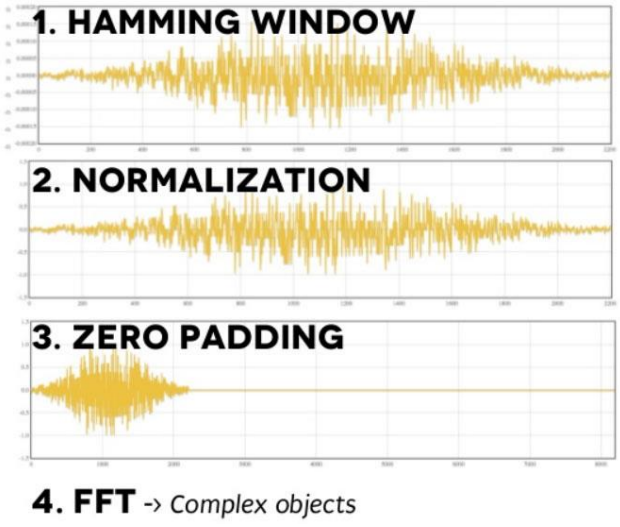


Figure 3.11 Τα βήματα του STFT [21]

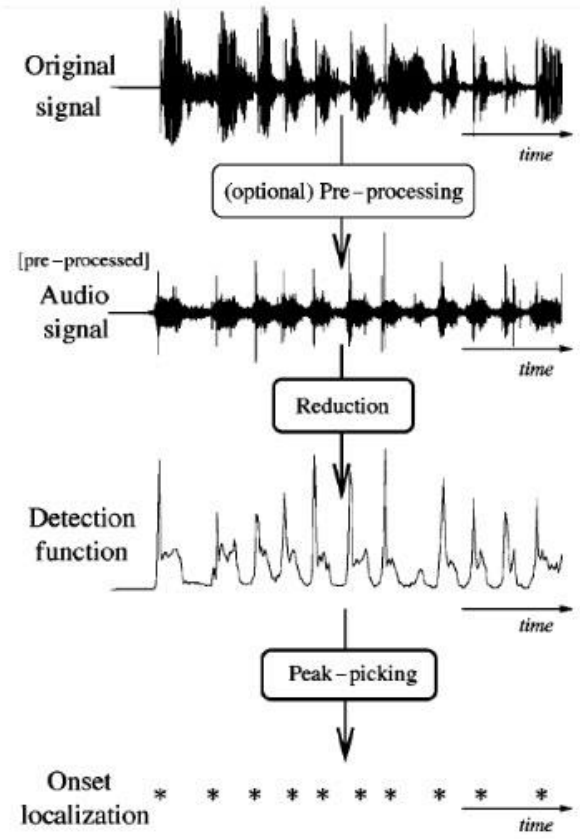


Figure 3.12 Διάγραμμα ροής ενός τυπικού αλγορίθμου εύρεσης των onsets [18]

3.1.3.2 Συναρτήσεις ανίχνευσης

Σε αυτή την υποενότητα θα γίνει μια σύντομη ανασκόπηση μερικών συναρτήσεων ανίχνευσης. Μία συνάρτηση ανίχνευσης onset συνήθως έχει χαμηλότερο ρυθμό δειγματοληψίας συγκριτικά με το ηχητικό σήμα, επιτυγχάνοντας με αυτό τον τρόπο μείωση δεδομένων χωρίς απώλεια της πληροφορίας των onset.

3.1.3.2.1 Συναρτήσεις μείωσης (*reduction*) στο πεδίο του χρόνου

Στην αναπαράσταση του πεδίου του χρόνου ενός ηχητικού σήματος, ένα onset αντιστοιχεί σε μία αύξηση του πλάτους του σήματος. Το 1985 ο Schloss πήρε αυτή την ιδέα και χρησιμοποίησε την περιβάλλουσα πλάτους του σήματος ως συνάρτηση ανίχνευσης:

$$E_0(n) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} |x(n+m)|w(m)$$

Όπου x η συνάρτηση ήχου και w(m) ένα ορθογώνιο παράθυρο N σημείων. Η συγκεκριμένη μεθοδολογία έχει ικανοποιητικά αποτελέσματα σε κρουστικά onsets.

3.1.3.2.2 Συναρτήσεις μείωσης (*reduction*) στο πεδίο του φάσματος

Για να μπορούν να εντοπιστούν onset εκτός των κρουστικών πρέπει να ληφθούν υπόψη αστάθειες στην σταθερή κατάσταση μίας απεικόνισης ενός ηχητικού σήματος στο πεδίο του φάσματος.

<u>Περιεχόμενο υψηλών συχνοτήτων (*High Frequency Content*)</u>

Ένα Onset έχει πιο έντονη ενέργεια στις ζώνες στις οποίες η ανάμειξη με άλλες παράλληλες συνιστώσες είναι μικρότερη, μια κατάσταση που συνήθως συμβαίνει σε υψηλής συχνότητας περιοχές. Το γεγονός αυτό μπορεί να αξιοποιηθεί σταθμίζοντας κάθε παράθυρο STFT με ένα παράγοντα ανάλογο της συχνότητας του. Ως εκ τούτου, αθροίζοντας όλα τα σταθμισμένα παράθυρα , λαμβάνουμε την συνάρτηση ανίχνευσης περιεχομένου υψηλών συχνοτήτων, παρουσιάζοντας την ως HFC ή E, ή οποία μπορεί να χρησιμοποιηθεί σαν συνάρτηση ανίχνευσης.

$$\widetilde{E_k}(n) = \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} W_k |X_k(n)|^2$$

Όπου Wk η στάθμιση εξαρτώμενη από τη συχνότητα.

Φασματική διαφορά (*Spectral Difference*)

Είναι δυνατόν να δημιουργήσουμε μία συνάρτηση ανίχνευσης η οποία αναπαριστά την «απόσταση» μεταξύ διαδοχικών STFT παραθύρων, αντιμετωπίζοντας τα σαν σημεία σε ένα N διαστάσεων χώρο. Αυτή η συνάρτηση ονομάζεται φασματική διαφορά (*spectral difference*) ή Spectral Flux (*SF*), μετρά την αλλαγή μεγέθους σε κάθε παράθυρο συχνοτήτων και υπολογίζεται από τη διαφορά δύο συνεχών φασμάτων μικρού χρόνου παράθυρο παράθυρο:

$$SF(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \{\mathrm{H}(|\mathrm{X_k}(\mathrm{n})| - |\mathrm{X_k}(\mathrm{n}-1)|)\}^2$$

$H(x) = \frac{x+|x|}{2}$ προσδιορίζουμε τη συνάρτηση ανόρθωσης μισού κύματος και έχει ως στόχο την εξάλειψη των αρνητικών διαφορών.

Απόκλιση Φάσης(*Phase Deviation*)

Τα μη κρουστικά onset είναι πιο εύκολα ανιχνεύσιμα εξετάζοντας τη φάση του πεδίου του φάσματος ενός ηχητικού σήματος, δεδομένου ότι ένα μεγάλο χρονικά μέρος της δομής του σήματος περιέχεται στη φάση του φάσματος. Η απόκλιση φάσης (*Phase Deviation ή PD*) ψάχνει για παρατυπίες στη φάση του φάσματος και ορίζεται ως εξής:

$$PD(n) = \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |\emptyset''_{\mathrm{k}}(\mathrm{n})|$$

Το Ø'' αναπαριστά την δεύτερη διαφορά της φάσης Ø.

Η μέθοδος απόκλισης φάσης θεωρεί όλες τις συχνότητες εξίσου ίσες, έτσι ο Dixon πρότεινε τη στάθμιση των παραθύρων συχνοτήτων σύμφωνα με το μέγεθος τους (magnitude) με σκοπό την απόκτηση μίας νέας συνάρτησης ανίχνευσης την οποία ονόμασε σταθμισμένη απόκλιση φάσης (*Weighted Phase Deviation ή WPD*):

$$WPD(n) = \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |\mathrm{X_k}(\mathrm{n})\emptyset''_{\mathrm{k}}(\mathrm{n})|$$

Όπου Xk(n) αναπαρηστάτε το μέγεθος(*magnitude*). [18]

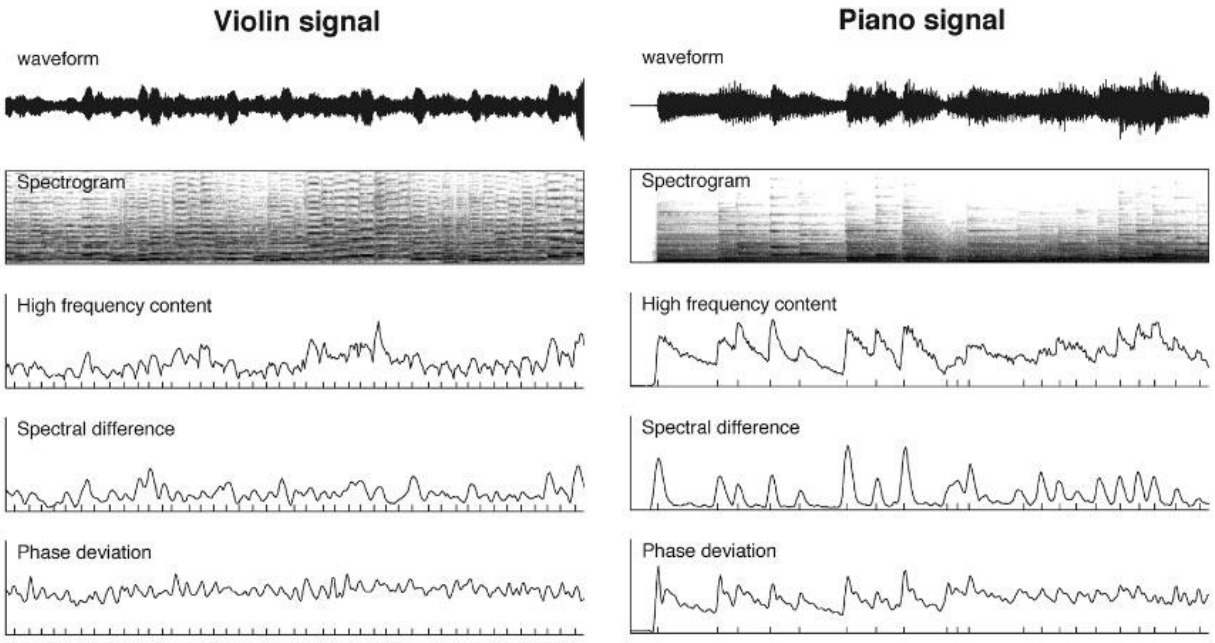


Figure 3.13 Συγκριτικές αναπαραστάσεις των μεθόδων ανίχνευσης στο πεδίο του χρόνου [22]

3.1.3.3 Επιλογή κορυφών

Μία συνάρτηση ανίχνευσης που δημιουργήθηκε με οποιονδήποτε από τους παραπάνω τρόπους συνήθως δείχνει ορθά τα τοπικά μέγιστα, γενικά με μία μεταβλητότητα εξαιτίας του θορύβου. Για να επιλέξουμε τα onset από μία συνάρτηση ανίχνευσης, αρκετά βήματα χρησιμοποιούνται που συνήθως εντάσσονται στις ακόλουθες κατηγορίες: Μετά-επεξεργασία (*Post-processing*), κατωφλίωση (*Thresholding*), και επιλογή κορυφών(*Peak-picking*)

Μετά-επεξεργασία (post-processing)

Όπως η προεπεξεργασία έτσι και η μετά-επεξεργασία είναι ένα προαιρετικό βήμα που εξαρτάτε από τη μέθοδο μείωσης που χρησιμοποιήθηκε για να δημιουργήσει τη συνάρτηση ανίχνευσης. Ο σκοπός της μετά-επεξεργασίας είναι να διευκολύνει τις διαδικασίες της κατωφλύωσης και της επιλογής κορυφών μέσω της αύξησης της ομοιομορφίας και της συνοχής των γεγονότων που σχετίζονται με τα χαρακτηριστικά του της συνάρτησης ανίχνευσης, ιδανικά μετατρέποντας τα σε απομονωμένα, εύκολα ανιχνεύσιμα τοπικά μέγιστα. Σε αυτή την κατηγορία εμπίπτουν οι διεργασίες που προορίζονται για μείωση των επιδράσεων του θορύβου (όπως ομαλοποίηση -

smoothing) και διεργασίες που είναι απαραίτητες για την επιτυχή επιλογή των παραμέτρων της κατωφλύωσης για ένα μεγάλο εύρος σημάτων (όπως κανονικοποίηση και αφαίρεση DC) [18]

Κατωφλύωση (*Thresholding*)

Η πρώτη προσέγγιση είναι να καθοριστεί ένα σταθερό κατώφλι(*threshold*) δ, και σε αυτή την περίπτωση τα onset θα είναι κορυφές όπου η συνάρτηση ανίχνευσης d είναι πάνω από το κατώφλι: d(n)≥δ

Από τη στιγμή που η μουσική επιδεικνύει μεγάλες δυναμικές (*dynamics*), τα σταθερά κατώφλια δίνατε να δίνουν περιορισμένης εγκυρότητας αποτελέσματα, έτσι είναι συνηθισμένο να χρησιμοποιούνται προσαρμοστικά κατώφλια (*adaptive thresholds*). Μία συνάρτηση προσαρμοστικού κατωφλιού, $\tilde{\delta}(n)$, μπορεί να δημιουργηθεί με πολλούς τρόπους, όμως ο πιο κοινός είναι η χρήση μίας κινούμενης συνάρτησης ενδιάμεσης τιμής:

$$\tilde{\delta}(n) = \delta + \lambda\, median(|d(n-M)|, \dots, |d(n+M)|)$$

Όπου λ και δ θετικές σταθερές, που μπορούν να ρυθμιστούν και M ένα ομαλό παράθυρο που μπορεί επίσης να ρυθμιστεί.

Το είδος της συνάρτησης κατωφλίωσης είναι ένα εύρωστο ζήτημα στις δυναμικές των σημάτων.
Είναι σημαντικό να παρατηρηθεί ότι οι παράμετροι που θα χρησιμοποιηθούν κατά την κατωφλίωση έχουν σημαντικό αντίκτυπο στα τελικά αποτελέσματα κυρίως στην αναλογία ψευδών θετικών (*false positives*) σε ψευδώς αρνητικών (*false negatives*). [22]

Peak-picking

Μετά τις παραπάνω διαδικασίες, η επιλογή των onset, o(n), περιορίζεται στην ταυτοποίηση των τοπικών μεγίστων που είναι πάνω από το ορισμένο κατώφλι, το οποίο μπορεί να συνοψιστεί σαν:

$$o(n) = \begin{Bmatrix} \mathbf{1}\ \boldsymbol{if}\ \boldsymbol{d(n) > \tilde{\delta}(n)} \\ \boldsymbol{and}\ \boldsymbol{d(n-1) \le d(n) \le d(n+1)} \\ \mathbf{0} \qquad \boldsymbol{otherwise} \end{Bmatrix}$$

3.1.4 Αυτόματη εύρεση ρυθμικής πληροφορίας (*Beat-tracking*)

Στη μουσική ορολογία, tempo είναι η ταχύτητα ή ο ρυθμός ενός συγκεκριμένου κομματιού ή μίας υποδιαίρεσης αυτού.

Μέτρηση tempo

Ένα κομμάτι του ρυθμού της μουσικής είναι συνήθως γραμμένο στην αρχή της παρτιτούρας, στη μοντέρνα δυτική μουσική συνήθως υποδηλώνεται σε χτύπους ανά λεπτό (*beats per minute - BPM*). Αυτό σημαίνει ότι μια συγκεκριμένη αξία νότας (για παράδειγμα ένα τέταρτο) ορίζεται ως χτύπος (beat) και ότι το χρονικό διάστημα μεταξύ διαδοχικών χτύπων είναι μία καθορισμένη υποδιαίρεση του λεπτού. Όσο περισσότεροι είναι οι χτύποι ανά λεπτό τόσο μικρότερος είναι ο χρόνος μεταξύ διαδοχικών χτύπων και έτσι ένα κομμάτι πρέπει να παιχτεί γρηγορότερα. Για παράδειγμα το tempo 60 BPM υποδηλώνει ένα χτύπο το δευτερόλεπτο, ενώ το tempo 120 BPM είναι δύο φορές γρηγορότερο, υποδηλώνοντας ένα χτύπο ανά 0.5 δευτερόλεπτα.

3.1.4.1 Εκτίμηση διαστήματος χτύπων (*Beat interval estimation*)

Αφού πάρουμε όλα τα αποτελέσματα από την ανίχνευση των onset θεωρούμε πως κάθε onset/κορυφή είναι μία θέση χτύπου (*beat*). Υπολογίζοντας την χρονική διαφορά των θέσεων κάθε χτύπου, μπορούμε να βρούμε N-1 διαστήματα χτύπων από N κορυφές.
Για παράδειγμα 4 κορυφές ανιχνευτήκαν. Οι θέσεις αυτών των κορυφών είναι 0 δευτερόλεπτα, 0.5 δευτερόλεπτα, 1.1 δευτερόλεπτα και 1.4 δευτερόλεπτα. Μπορούμε να υπολογίζουμε τη χρονική διαφορά αφαιρώντας από τα γειτονικά. Τα διαστήματα χτύπων αυτών των κορυφών είναι 0.5 του δευτερολέπτου, 0.6 του δευτερολέπτου και 0.3 του δευτερολέπτου.

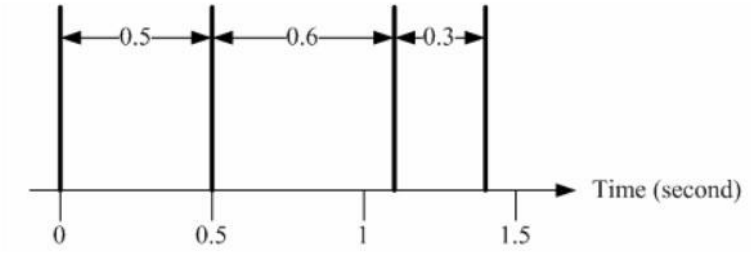


Figure 3.14 Τρία διαστήματα που υπολογίζονται απο τέσσερις κορυφές [23]

3.1.4.2 Ιστόγραμμα των διαστημάτων των χτύπων

Συλλέγοντας όλα τα διαστήματα (*beat intervals*), μπορούμε να δημιουργήσουμε ένα ιστόγραμμα για να αναλύσουμε τα δεδομένα. Κάθε διάστημα προστίθεται στο ιστόγραμμα μετρώντας τον αριθμό εμφανίσεων του. Το διάστημα που έχει τις περισσότερες εμφανίσεις είναι πολύ πιθανό να είναι το tempo των εισαγόμενων δεδομένων.

Για παράδειγμα αν τα περισσότερα συμβάντα που καταγράφονται είναι διαστήματος 0.478 του δευτερολέπτου μπορούμε να ακολουθήσουμε την ακόλουθη εξίσωση για να μετατρέψουμε το διάστημα χτύπων σε χτύπους ανά λεπτό (*BPM*).

$$BPM = 60 / \text{beat interval}$$

Έτσι το tempo γι αυτό το διάστημα είναι 60 / 0.478 = 125.5 BPM

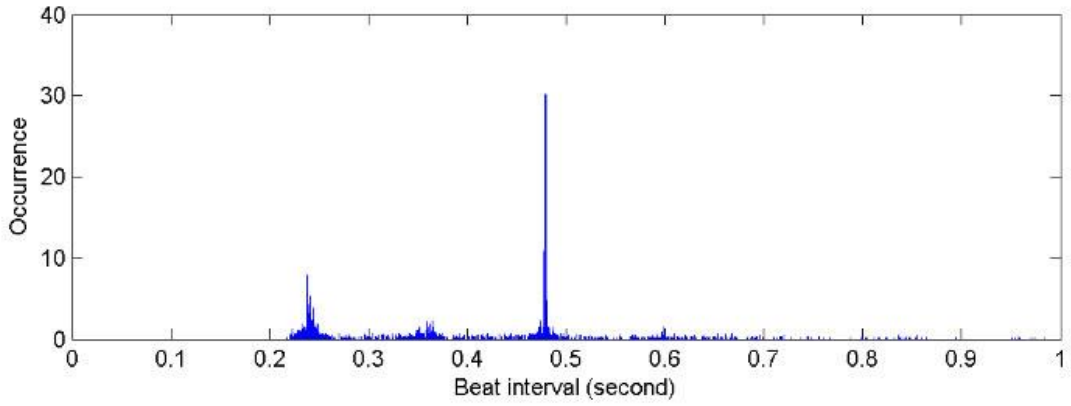


Figure 3.15 Ιστόγραμμα διαστημάτων [23]

3.1.4.3 Βελτίωση του ιστογράμματος

Χρησιμοποιώντας το παραπάνω ιστόγραμμα απευθείας, η απόδοση της εκτίμησης tempo μπορεί να μην είναι τόσο καλή. Για να βελτιώσουμε την απόδοση του συστήματος, πρέπει να βελτιώσουμε το ιστόγραμμα ώστε να έχουμε μεγαλύτερη ακρίβεια.

3.1.4.4 Εισάγοντας βαρύτητα στα περιστατικά των διαστημάτων

Μπορούμε να διεργαστούμε τα περιστατικά των διαστημάτων με κάποια βαρύτητα για να αυξήσουμε την ευστοχία του συστήματος. Υπάρχουν πολλές μέθοδοι εισαγωγής βαρύτητας στα περιστατικά των διαστημάτων. Πρέπει να ορίσουμε το βάρος με λογικό τρόπο για να αυξήσουμε την απόδοση. Για παράδειγμα, παίρνουμε ένα beat interval που λέγεται 'B', αν όλοι οι γείτονες έχουν τις ίδιες τιμές με το 'B' τότε το βάρος του 'B' πρέπει να είναι πολύ υψηλό. Ομοίως αν όλοι οι γείτονες έχουν διαφορετικές τιμές από το 'B' το βάρος πρέπει να είναι πολύ μικρό. Έτσι μπορούμε να ορίσουμε το βάρος του 'B' σύμφωνα με το πόσοι γείτονες του 'B' έχουν όμοιες τιμές.
Υπάρχουν δύο παράμετροι που μπορούμε να ορίσουμε. Η πρώτη είναι πόσους γείτονες πρέπει να συγκρίνουμε και η δεύτερη σαφήνεια της ομοιότητας των τιμών.
Παραδείγματος χάρη για την πρώτη παράμετρο ένα σύστημα μπορεί να χρησιμοποιεί τέσσερα διαστήματα σε κάθε πλευρά του 'B'. Επομένως υπάρχουν οκτώ διαστήματα προς σύγκριση. Αν χρησιμοποιήσουμε πάρα πολλούς γείτονες στη σύγκριση το βάρος μπορεί να μην είναι ακριβές επειδή οι μακρινοί γείτονες μπορείς να παίρνουν διαφορετική τιμή. Αν ο αριθμός των διαστημάτων είναι πολύ μικρός η βαρύτητα δεν είναι τόσο χρήσιμη.

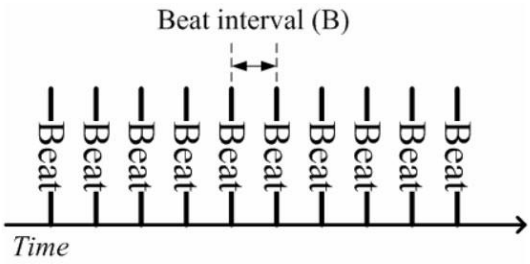


Figure 3.16 Οκτώ διαστήματα χρησιμοποιούνται για σύγκριση [23]

Για τη δεύτερη παράμετρο, η επιτρεπόμενη απόκλιση ΔΟΠ (*IOI deviation*) πρέπει να είναι μία τιμή σχετική με το tempo του κομματιού. Μία προτεινόμενη τιμή μπορεί να υπολογιστεί από τη φόρμουλα y=320.67x -0.3388, όπου χ είναι το tempo του κομματιού και y η προτεινόμενη επιτρεπόμενη τιμή απόκλισης ΔΟΠ. Ωστόσο δεν ξέρουμε πόσο είναι το tempo του κομματιού σε αυτό το σημείο. Έτσι χρησιμοποιούμε την τιμή του διαστήματος 'B' για να υπολογίσουμε τη φόρμουλα x = 60 / B. Μετά μπορούμε να πάρουμε την τιμή του y χρησιμοποιώντας το παραπάνω x. Με την χρήση της τιμής του y μπορούμε να ορίσουμε ότι δύο αποκλίσεις ΔΟΠ είναι ίδιες αν η διαφορά μεταξύ του y και της τιμής του διαστήματος χτύπου 'B' είναι μικρότερη από το y.

Αφού ρυθμίσουμε αυτές τις δύο παραμέτρους, υπολογίζουμε το βάρος μετρώντας τον αριθμό των κοινών διαστημάτων μεταξύ εννέα διαστημάτων.

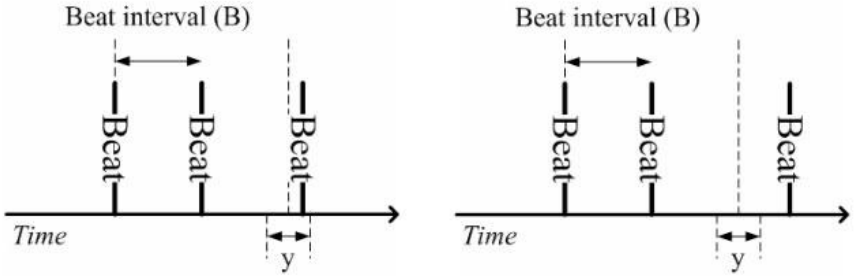


Figure 3.17 Όμοιο beat αριστερά και ανόμοιο δεξιά [23]

3.1.4.5 Εξομαλύνοντας το ιστόγραμμα (*smoothing*)

Μερικές φορές μπορεί να συναντήσουμε κάποιες ιδιαίτερες περιπτώσεις στο ιστόγραμμα. Εξετάζοντας ένα παράδειγμα όπου το ιστόγραμμα εμφανίζει πολλαπλά περιστατικά μέγιστης βαρύτητας παρατηρούμε ότι πρέπει να επιλέξουμε ένα συγκεκριμένο tempo από πολλά συμβάντα μέγιστης βαρύτητας. Εξετάζοντας μία άλλη περίπτωση όπου το συμβάν μέγιστης βαρύτητας προκύπτει σε χαμηλής πυκνότητας περιοχή, μπορούμε απλά να το επιλέξουμε και να υπολογίσουμε το tempo βάση αυτού. Ωστόσο αυτό το μέγιστο δεν είναι η πλειοψηφία στην κατανομή (*distribution*) των διαστημάτων. Η κύρια περιοχή είναι γύρω στα 0.4 δευτερόλεπτα αλλά όχι στα 0.8. Έτσι μπορεί να μην είναι τόσο ορθό απλά να επιλέξουμε το μέγιστης βαρύτητας συμβάν χωρίς να λάβουμε υπόψη μας την κατανομή.

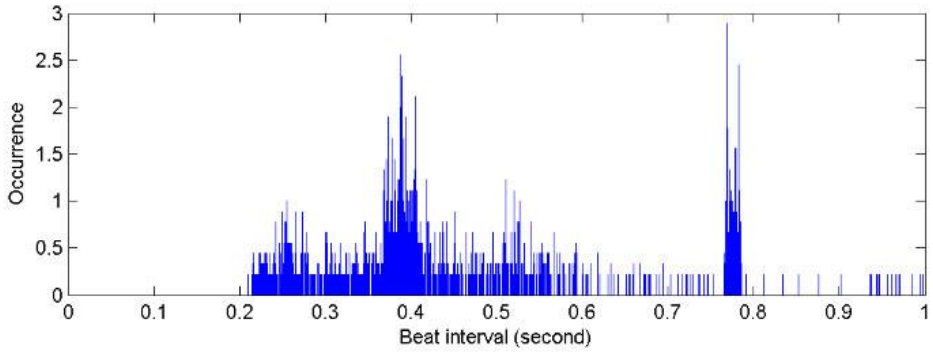


Figure 3.18 Ένα ιστόγραμμα διαστημάτων στο οποίο το μέγιστο προκύπτει στην περιοχή χαμηλής πυκνότητας διαστημάτων [23]

Μπορούμε να ξεπεράσουμε τα παραπάνω προβλήματα με μία λύση: Εξομαλύνοντας το ιστόγραμμα πολλαπλασιάζοντας το κάθε στοιχείο με μία συνάρτηση Gaussian. Αφού το ιστόγραμμα εξομαλυνθεί μπορούμε να τα διαστήματα από το μέγιστο πλάτος.

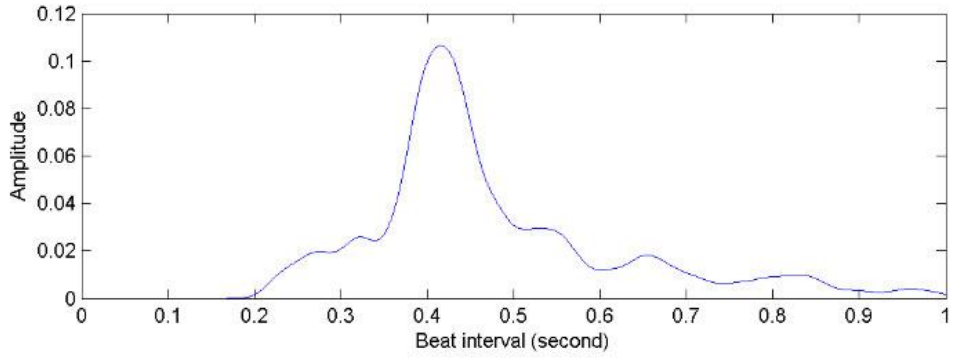


Figure 3.19 Εξομαλυμένο ιστόγραμμα του πρώτου παραδείγματος [23]

Τα πολλαπλά συμβάντα μέγιστης βαρύτητας εξαφανίστηκαν. Έτσι τώρα μπορούμε να πάρουμε το μέγιστο πλάτος για να υπολογίσουμε το tempo. Για το παράδειγμα μας το διάστημα είναι 0.416 δευτερόλεπτα στο μέγιστο πλάτος. Οπότε το tempo είναι 144.2 BPM.

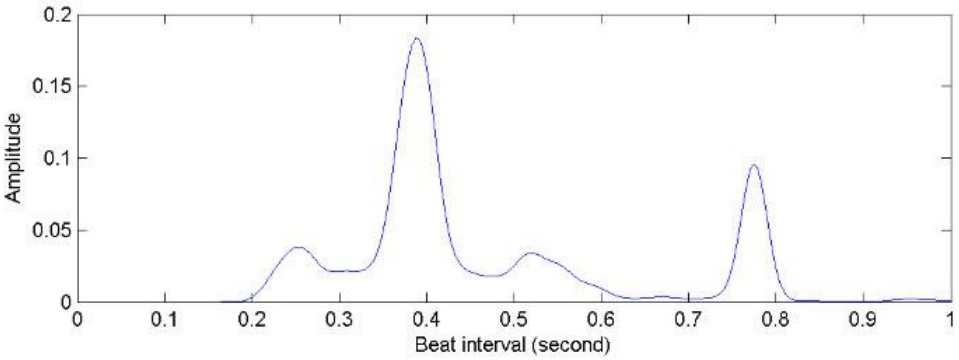


Figure 3.20 Εξομαλυμένο ιστόγραμμα του δεύτερου παραδείγματος [23]

Επίσης το αυθεντικό συμβάν μέγιστης βαρύτητας δεν είναι πια το μέγιστο πλάτος. Αυτό συμβαίνει γιατί η συνάρτηση Gaussian χρησιμοποιεί την κατανομή των διαστημάτων. Το πλάτος στη μείζονα περιοχή θα αυξηθεί ενώ θα μειωθεί στην ελάσσονα. Γι αυτό το παράδειγμα το διάστημα είναι 0.389 του δευτερολέπτου στο μέγιστο πλάτος. Έτσι το tempo είναι 154.2 BPM.

3.1.4.6 Μέγιστη τιμή εμπιστοσύνης(*Maximum confidence value*)

Μπορούμε πρώτα να ορίσουμε μία τιμή εμπιστοσύνης για ένα ιστόγραμμα. Το σύστημα την ορίζει σαν την περιοχή υπό της κορυφής με μήκος 0.1 του δευτερολέπτου σε όλη την περιοχή κάτω από την καμπύλη. Χρησιμοποιούμε την τιμή 0.1 γιατί είναι το ίδιο μήκος με τη 3SD ροής Gaussian συνάρτησης. Επομένως κάθε ιστόγραμμα θα πάρει μία τιμή εμπιστοσύνης. Τελικά, το σύστημα μας θα επιλέξει το tempo με ένα τέστ τιμής εμπιστοσύυνης η οποία αποκτά το τελικό tempo παίρνοντας την μέγιστη τιμή εμπιστοσύνης. [23]

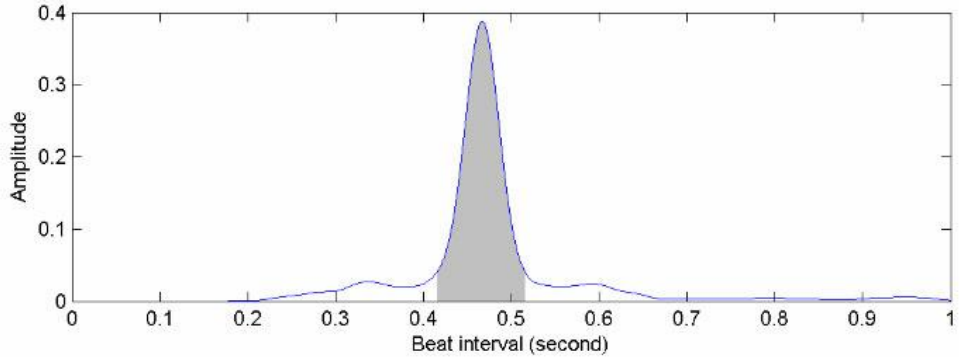


Figure 3.21 Η τιμή εμπιστοσύνης ορίζεται ως η περιοχή κάτω από την κορυφή με 0,1 δευτερόλεπτα πλάτος (γκρι χρώμα) σε όλη την περιοχή κάτω από την καμπύλη [23]

3.1.5 Εκτίμηση του τονικού ύψους (*pitch*) της νότας

Το τονικό ύψος (*pitch*) ορίζεται σαν ιδιότητα ενός ακουστικού σήματος το οποίο καθορίζεται από τη συχνότητα των κυμάτων που το παράγουν: έτσι το τονικό ύψος αναπαριστά το πόσο ψηλός ή χαμηλός είναι ο ήχος. Αυστηρά μιλώντας, ο όρος pitch θα πρέπει να θεωρηθεί ως η ακουστική αντίληψη του τόνου. Το τονικό ύψος είναι εκ φύσεως υποκειμενική ποσότητα και δεν μπορεί να μετρηθεί απευθείας από το ακουστικό σήμα. Είναι μία μη γραμμική συνάρτηση του φάσματος και της χρονικής κατανομής της ενέργειας του σήματος. Η θεμελιώδης συχνότητα (*F0*) είναι η ποσότητα που μετριέται σχεδόν από όλους τους αλγόριθμους εκτίμησης τονικού ύψους. Η F0 ορίζεται ως η αντίστροφη της περιόδου του σήματος. Το αντιληπτό τονικό ύψος ενός σήματος είναι υψηλά συσχετιζόμενο με την θεμελιώδη του συχνότητα. Η εκτίμηση του τονικού ύψους είναι ζωτικής σημασίας για πολλές εφαρμογές επεξεργασίας ήχου και φωνής. Για τη φωνή, η εκτίμηση τονικού ύψους εντοπίζει το περίγραμμα του τονικού ύψους της ομιλίας, και ως εκ τούτου λειτουργεί σαν ένα σημαντικό μέρος των αλγορίθμων αναγνώρισης

φωνής. Για τη μουσική, η εκτίμηση τονικού ύψους αντιστοιχεί τις αναγνωρισθέντες συχνότητες σε ορισμένες μουσικές νότες. Πολλοί αλγόριθμοι εκτίμησης ύψους έχουν αναπτυχθεί. Παρόλα αυτά, καμία από τις υπάρχουσες μεθόδους δεν μπορεί να εγγυηθεί εύστοχη αναγνώριση όταν υπάρχουν υψηλές στάθμες θορύβου στο σήμα.

Οι αλγόριθμοι εκτίμησης τονικού ύψους που χρησιμοποιούνται συνήθως περιλαμβάνουν τρία στάδια:

1. Προ-επεξεργασία του ακουστικού σήματος.
2. Δημιουργία πιθανών υποψήφιων pitch.
3. Μετά-επεξεργασία για να επιλεγεί η βέλτιστη εκδοχή ανάμεσα στις υποψηφιότητες ώστε να βελτιστοποιηθεί η εκτίμηση της F0.

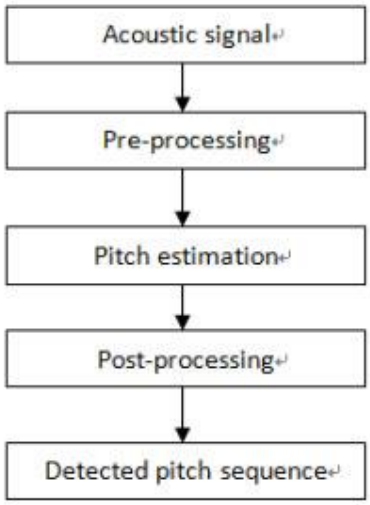


Figure 3.22 Η ροή ενός τυπικού αλγορίθμου εκτίμησης τονικού ύψους [24]

3.1.5.1 Προ-επεξεργασία (*Pre-processing*)

Ο σκοπός της προ-επεξεργασίας είναι να εξαλείψει τις συνιστώσες σημάτων παρεμβολής.

Φιλτράρισμα (*filtering*)

Το φιλτράρισμα είναι η πιο κοινή τεχνική που χρησιμοποιείται στο στάδιο της προ-επεξεργασίας. Για παράδειγμα το εύρος του τονικού ύψους του λόγου είναι 50Hz με

600Hz. Ένα ζωνοπερατό (*bandpass*) φίλτρο μπορεί αποτελεσματικά να αφαιρέσει τις συχνότητες έξω από το επιθυμητό εύρος ζώνης, το οποίο βελτιώνει την εκτίμηση της F0.

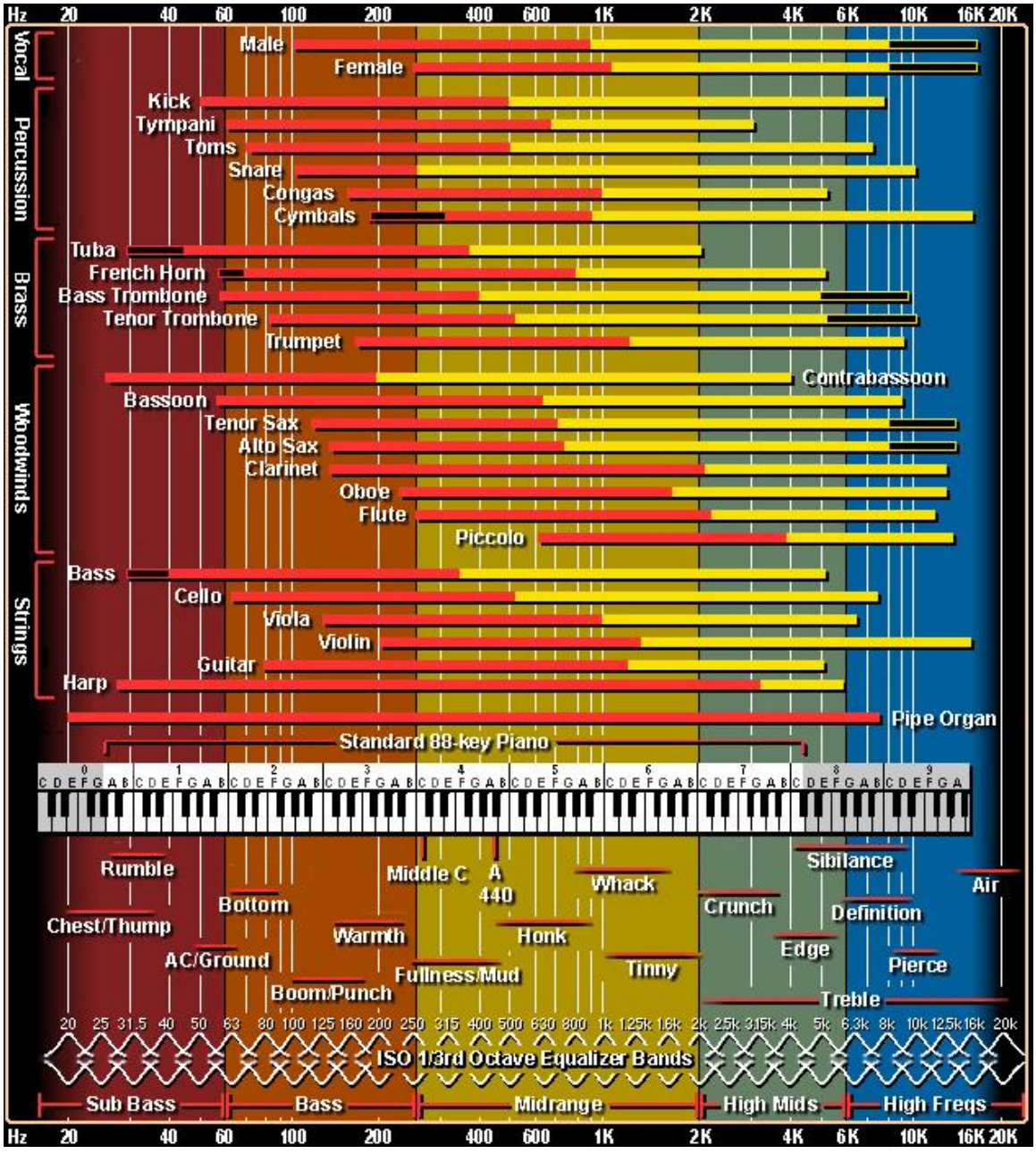


Figure 3.23 Εύρος συχνοτήτων ανά τύπο οργάνου [25]

Μη-γραμμικές λειτουργίες (*Non-linear operations*)

Μία μη γραμμική τεχνική που συνήθως χρησιμοποιείται είναι ο ορισμός ενός κατωφλιού (*threshold*) ώστε να αφαιρεθεί ο θόρυβος χαμηλού επιπέδου. Μία άλλη κοινή πρακτική είναι το κέντρο-αποκοπής (*center-clipping*)

Η συνάρτηση κέντρου αποκοπής ορίζεται ακολούθως:

$$y(n) = clc[x(n)] = \begin{cases} (x(n) - C_L) & x(n) \geq C_L \\ 0 & |x(n)| < C_L \\ (x(n) + C_L) & x(n) \leq C_L \end{cases}$$

Όπου C_L είναι το κατώφλι ψαλιδισμού, το οποίο γενικώς είναι περίπου το 50% της μέγιστης απόλυτης τιμής του σήματος εισόδου. Με αυτή την τεχνική οι επιθυμητές κορυφές γίνονται πιο καθαρές, ενώ οι άλλες άσχετες κορυφές ακλουθούν.

3.1.5.2 Δημιουργία υποψηφίων τονικών υψών

Οι αλγόριθμοι εκτίμησης τονικού ύψους μπορούν να χωριστούν σε τρείς κατηγορίες:

- Προσεγγίσεις στο πεδίο του χρόνου (*time domain*)
- Προσεγγίσεις στο πεδίο της συχνότητας (*frequency domain*)
- Στατιστικές προσεγγίσεις

3.1.5.2.1 Προσεγγίσεις στο πεδίο του χρόνου (*time domain*)

Αναλογία μηδενικής διέλευσης (*Zero-crossing Rate*)

Η αναλογία μηδενικής διέλευσης (*ZCR*) αναπαριστά την αναλογία εισερχόμενων αλλαγών στο σήμα (π.χ. την αναλογία στην οποία το σήμα αλλάζει από θετικό σε αρνητικό κα αντίστροφα). Αυτή η τεχνική είναι πολύ απλή με ελάχιστες υπολογιστικές απαιτήσεις. Η αναλογία μηδενικής διέλευσης ορίζεται ως:

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1} \Pi\{S_t S_t + 1 < 0\}$$

Όπου S_t είναι η τιμή του σήματος τη στιγμή t, και Π{A} είναι 1 αν το A είναι αληθές και 0 αν είναι ψευδές. Η συχνότητα του σήματος F_{signal} δίνεται από:

$$F_{signal} = \frac{zcr + fs}{2}$$

Ωστόσο κάτω από θορυβώδης συνθήκες, η επίδοση αυτής της τεχνικής είναι πολύ φτωχή επειδή ο ήχος σε χαμηλά επίπεδα μπορεί εύκολα να αλλάξει σημεία του σήματος.

Αυτοσυσχέτιση (Autocorrelation)

Η αυτοσυσχέτιση, επίσης γνωστή ως σειριακή συσχέτιση (*serial correlation*) ή διασταυρούμενη αυτοσυσχέτιση (*cross-autocorrelation*), είναι η διασταυρούμενη συσχέτιση ενός σήματος με τον εαυτό του σε διαφορετικά χρονικά σημεία. Ανεπίσημα, είναι η ομοιότητα μεταξύ παρατηρήσεων ως συνάρτηση του χρόνου που μεσολαβεί μεταξύ τους. Είναι ένα μαθηματικό εργαλείο για την εύρεση επαναλαμβανόμενων μοτίβων, όπως η παρουσία ενός περιοδικού σήματος που επισκιάζεται από θόρυβο ή η αναγνώριση της απούσας θεμελιώδους συχνότητας σε ένα σήμα που συνάγεται από τις αρμονικές συχνότητες της. [26]

Η συνάρτηση της αυτοσυσχέτισης r(τ) ενός σήματος με χρονική καθυστέρηση τ ορίζεται σαν:

$$r(\tau) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x(n+\tau)$$

Η συνάρτηση αυτοσυσχέτισης έχει πάντα ένα καθολικό μέγιστο (global maximum) για τ = 0. Αν το σήμα είναι περιοδικό, η συνάρτηση πρέπει να έχει καθολικά μέγιστα στα πολλαπλάσια της περιόδου του σήματος T_0 τέτοια ώστε $r_{x(T_0)=r_x(0),n=1,2,3\ldots}$ Στην πράξη το x(t) είναι συνήθως ένα μη περιοδικό παραθυριοποιημένο σήμα. Εντεύθεν κανένα καθολικό μέγιστο δεν μπορεί να βρεθεί έξω από το τ=0. Ωστόσο, μπορεί να υπάρχουν ακόμη κάποια τοπικά μέγιστα. Εάν το υψηλότερο των τοπικών μεγίστων είναι σε μια χρονική καθυστέρηση τα και η τιμή σε αυτό το σημείο είναι μεγαλύτερη από το κατώφλι το σήμα λέγεται ότι έχει ένα περιοδικό μέρος. Η θεμελιώδης συχνότητα F0 εκτιμάτε ως 1/τ.

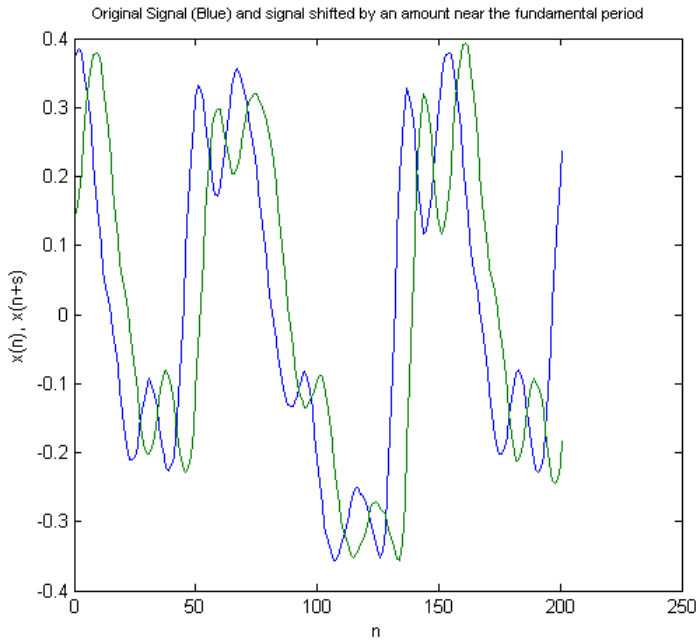


Figure 3.24 Μετατοπίζοντας το σήμα [27]

YIN

Ο αλγόριθμος YIN χρησιμοποιεί μία συνάρτηση διαφοράς (*difference function*) βασισμένη στη μέθοδο της αυτοσυσχέτησης. Ενώ η λειτουργία της αυτοσυσχέτισης στοχεύει στο να μεγιστοποιεί το γινόμενο μεταξύ της κυματομορφής και της μετατοπισμένης της εκδοχής, η συνάρτηση διαφοράς $d_t(\tau)$ στοχεύει στο να ελαχιστοποιεί τη διαφορά μεταξύ της κυματομορφής και της μετατοπισμένης της εκδοχής.

$$d_t(\tau) = \sum_{j=1}^{W} (x_j - x_{j+\tau})^2$$

Όπου W είναι το μέγεθος του παραθύρου.
Για να χειριστεί τη σχεδόν φυσική περιοδικότητα του τονικού ύψους σε πραγματικά σήματα, ο αλγόριθμος YIN κανονικοποιεί τη συνάρτηση διαφοράς βάση της αθροιστικής μέσης τιμής του και ορίζει τιμή 1 για τ = 0:

$$d'_t(\tau) = \begin{cases} 1 & if\ \tau = 0 \\ d_t(\tau)/[(1/\tau)\sum_{j=1}^{\tau} d_t(j)] & otherwise \end{cases}$$

Τα τελευταία τρία βήματα περιλαμβάνουν την τοποθέτηση ενός ορίου στην μικρότερη τιμή του τ η οποία είναι αποδεκτή. Επίσης, χρησιμοποιείται παραβολική παρεμβολή (parabolic interpolation) για να βελτιώσει τη θέση της κορυφής και γίνεται αναζήτηση γύρω από του δείκτες του τονικού ύψους για να βελτιωθεί περαιτέρω η εκτίμηση.

Praat

Η βασική αρχή πίσω από τον αλγόριθμο Praat είναι η μέθοδος αυτοσυσχέτισης. Ένα μικρό τμήμα του κανονικοποιημένου σήματος x(t) κατ αρχήν πολλαπλασιάζεται με ένα (Hamming, Hanning) παράθυρο w (t) το οποίο έχει αποτέλεσμα $r_a(\tau)$. Η μέθοδος της αυτοσυσχέτησης $r_a(\tau)$ υπολογίζεται από:

$$F0 = \frac{1}{time\ lag\ of\ the\ maximum\ peak\ \times\ sampling\ frequency}$$

Ωστόσο αν χρησιμοποιήσουμε την μέθοδο επιλογής κορυφών που συστήνεται από την μέθοδο αυτοσυσχέτησης, μία λανθασμένη κορυφή θα επιλεχθεί. Για διαχειριστεί το πρόβλημα η συνάρτηση αυτοσυσχέτησης $r_a(t)$ διαιρείτε με $r_\omega(\tau)$. Κατόπιν η χρονική καθυστέρηση T_0 της μέγιστης κορυφής εκτιμάτε ότι είναι η περίοδος του x(t).

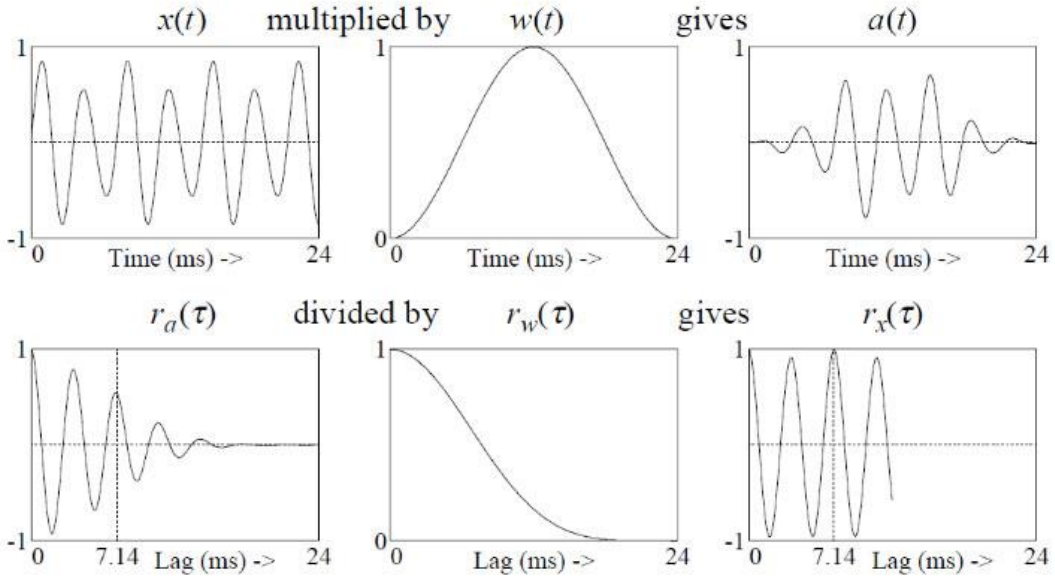


Figure 3.25 Διάγραμμα ροής του αλγορίθμου Praat [27]

3.1.5.2.2 Προσεγγίσεις στο πεδίο της συχνότητας (*frequency domain*)

Φάσμα αρμονικών γινομένων (*Harmonic Product Spectrum - HPS*)

Η HPS είναι μία κλασσική μέθοδος εντοπισού του τονικού ύψους βασισμένη στο γεγονός ότι οι κορυφές του φάσματος συχνοτήτων βρίσκονται στα πολλαπλάσια της θεμελιώδους συχνότητας. Στη μέθοδο αυτή μειώνεται η δειγματοληψία του αυθεντικού φάσματος συχνοτήτων υπό N(N=2,3,4…) και μετά όλα αυτά τα φάσματα πολλαπλασιάζονται μεταξύ τους. Η μέγιστη κορυφή δείχνει τη θεμελιώδη συχνότητα.

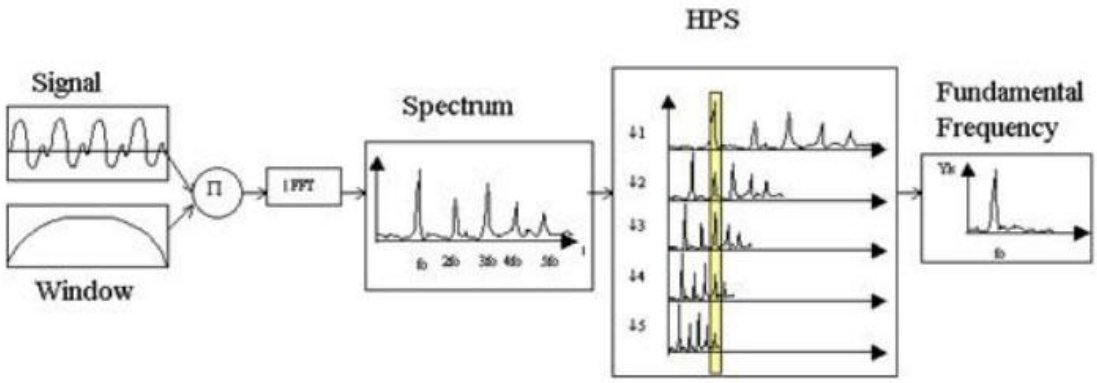


Figure 3.26 Διάγραμμα ροής του αλγορίθμου HPS [27]

Cepstrum

Το Cepstrum ορίζεται ως το αντίστροφο DFT της λογαριθμικής έκτασης του DFT του σήματος.

$$C(n) = F^{-1}\log |F(x[n])$$

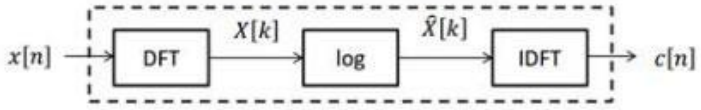


Figure 3.27 Διάγραμμα ροής του αλγορίθμου Cepstrum [27]

Το Cepstrum τείνει να έχει τοπικά μέγιστα στα K*T που αντιστοιχούν σε ακέραια πολλαπλάσια της γλωττιδικής περιόδου (*glottal period*). Ο λογαριθμικός φορέας της

έκτασης του φάσματος τείνει να ακολουθεί τις αρμονικές κορυφές στο φάσμα και ως εκ τούτου οδηγεί σε πιο διακριτές περιοδικές κορυφές στο cepstrum. Η ανεξάρτητη μεταβλητή ενός γραφήματος cepstrum ονομάζεται quefrency. Quefrency είναι ένα μέτρο του χρόνου, αν και όχι με την έννοια ενός σήματος στο πεδίο του χρόνου. Για παράδειγμα αν ο ρυθμός δειγματοληψίας ενός ακουστικού σήματος είναι 44,100 Hz και υπάρχει μεγάλος αριθμός κορυφών στο cepstrum των οποίων η quefrency είναι 100 δείγματα, η κορυφή που υποδηλώνει την ύπαρξη τονικού ύψους το οποίο είναι 44100/100 =441 Hz.

BaNa

Ο αλγόριθμος εκτίμησης τονικού ύψους BaNa είναι πρόσφατα ανεπτυγμένη μέθοδος που συνδυάζει την ιδέα εύρεσης κορυφών και Cepstrum για την επιλογή υποψήφιων F0. Ο BaNa επίσης εισαγάγει την ιδέα μίας βαθμολογίας εμπιστοσύνης για την επιλογή ενός τονικού ύψους μεταξύ πολλών υποψηφίων και ενσωματώνει τον αλγόριθμο Viterbi στη μετά-επεξεργασία για να εξαφανίσει κάποιες μη επιθυμητές υποψηφιότητες.

Ο αλγόριθμος αρχικά αναζητάει για αρμονικές κορυφές πάνω από ένα ορισμένο κατώφλι και μετά επιλέγει τις πέντε κορυφές από το σύνολο που έχουν τις χαμηλότερες συχνότητες. Ύστερα υπολογίζει την αναλογία των συχνοτήτων για κάθε δύο αρμονικές κορυφές. Εάν οποιαδήποτε υπολογισμένη αναλογία εντάσσεται στο εύρος ανοχής των αρμονικών αναλογιών που φαίνονται στον πίνακα … ένα υποψήφιο δυνητικό τονικό ύψος $\tilde{F}$ μπορεί να ληφθεί διαιρώντας την αναλογία της αρμονικής $\tilde{F}$ σε F0: $\tilde{F}=\tilde{F}$ /m. Η ιδέα επιτρεπόμενου εύρους εμπνεύστηκε από το γεγονός ότι οι αναλογίες των συχνοτήτων των αρμονικών κορυφών δεν είναι πάντα ακέραιοι.

Επιπλέον, η τιμή τονικού ύψους που βρέθηκε χρησιμοποιώντας τον αλγόριθμο Cepstrum περιλαμβάνεται επίσης ως υποψήφιο τονικό ύψος. Ο αλγόριθμος BaNa επίσης έχει μία παραλλαγή που ονομάζεται BaNa music, η οποία ειδικεύεται στον εντοπισμό τονικού ύψους σε μουσική. Ο BaNa music εκμεταλλεύεται το γεγονός ότι οι αρμονικές της μουσικής έχουν ένα ευρύτερο φάσμα συχνοτήτων έτσι αντί να χρησιμοποιεί ένα κατώφλι πλάτους και επιλέγοντας τις πέντε κορυφές με τη χαμηλότερη συχνότητα με ένα πλάτος πάνω από το κατώφλι, Οι πέντε αρμονικές κορυφές με τα υψηλότερα πλάτη επιλέγονται.

3.1.5.3 Μετά-επεξεργασία

Μερικές φορές εμφανίζονται λάθη στην επιλογή του τονικού ύψους ενός δεδομένου πλαισίου του σήματος. Τα πιο κοινά από αυτά τα λάθη είναι ένας διπλασιασμός ή

μείωση κατά το ήμισυ της θεμελιώδους περιόδου. Η μετά-επεξεργασία μπορεί να βοηθήσει στην αντιστάθμιση της επίδρασης τέτοιων λαθών. Κοινές πρακτικές συμπεριλαμβανομένης της εξομάλυνσης (*smoothing*) και του δυναμικού προγραμματισμού. [27]

3.2 Εφαρμογές ανάκτησης μουσικής πληροφορίας

Ableton Live 9 (*https://www.ableton.com/*)

Το ableton live είναι ένα ευρείας χρήσεως πλήρες desktop λογισμικό επεξεργασίας ψηφιακού ήχου. Στην τελευταία του έκδοση εισαγάγει μία σειρά αλγορίθμων οι οποίοι χρησιμεύουν στην ανάκτηση μουσικής πληροφορίας από ένα ηχητικό σήμα και στη αποτύπωση της σε αναπαράσταση MIDI. Πιο αναλυτικά, προσφέρει δύο λειτουργίες οι οποίες διαχωρίζονται ανάλογα με το αν το μουσικό κομμάτι προς ανάλυση είναι μονοφωνικό, δηλαδή αν δεν αναπαράγονται πολλές νότες ταυτόχρονα, ή πολυφωνικό, δηλαδή μπορεί να γίνει και αναγνώριση συγχορδιών. Η ανάκτηση της μουσικής πληροφορίας δεν γίνεται σε πραγματικό χρόνο, αλλά κατ 'απαίτηση του χρήστη. Η εφαρμογή εξάγει σχετικά ακριβή αποτελέσματα.

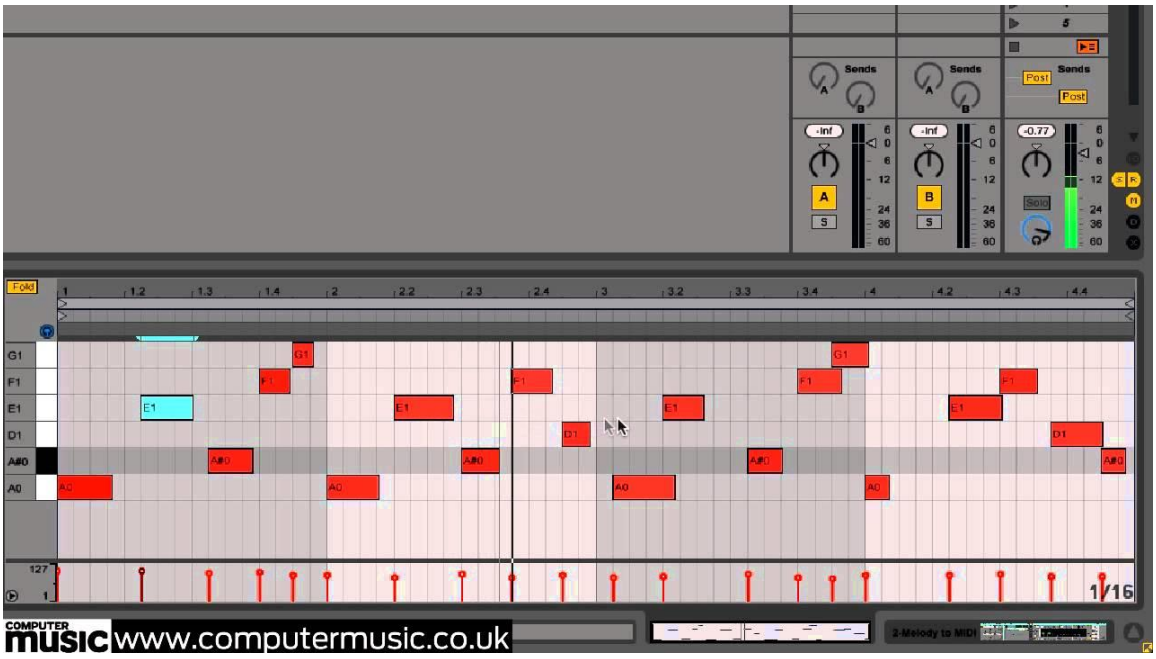


Figure 3.28 Εξαγωγή μελωδίας σε περιβάλλον MIDI στο Ableton live 9

AudioScore / Sibelius (*http://www.sibelius.com/products/audioscore/ultimate.html*)

Το Sibelius είναι ένα λογισμικό καταγραφής παρτιτούρας (*scorewriter*), χρησιμοποιείται από συνθέτες, ενορχηστρωτές, ερμηνευτές, μουσικούς εκδότες, εκπαιδευτικούς και μαθητές. Στο Sibelius συμπεριλαμβάνεται ένα λογισμικό καταγραφής μουσικής πληροφορίας το οποίο ονομάζεται AudioScore. Με αυτό ο χρήστης μπορεί να μετατρέψει ένα αρχείο ήχου σε παρτιτούρα ή ακόμα και να βλέπει αυτό που αναπαράγει μπροστά στο μικρόφωνο απεικονισμένο σε περιβάλλον MIDI σε πραγματικό χρόνο. Μπορεί να αναγνωρίσει μέχρι δύο νότες που αναπαράγονται ταυτόχρονα, δηλαδή υποστηρίζει μονοφωνική αναγνώριση και μπορεί να αντιληφθεί νότες μεγαλύτερης διάρκειας από 1/16. Τα αποτελέσματα έχουν ένα μέτριο βαθμό εγγυρότητας.

Figure 3.29 Το περιβάλλον του Audioscore κατά τη διάρκεια καταγραφής.

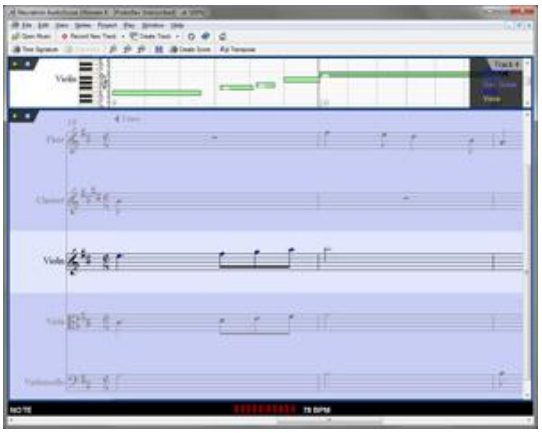


Figure 3.30 Το περιβάλλον του Audioscore κατά τη διάρκεια αναπαραγωγής MIDI ταυτόχρονα με παρτιτούρα

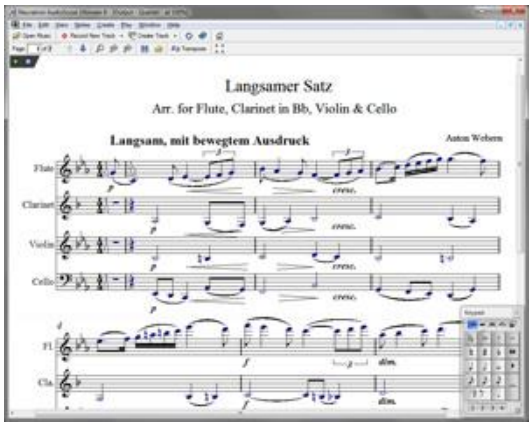


Figure 3.31 Το περιβάλλον μουσικής σημειογραφίας Sibelius

3.2.1 Εφαρμογές που χρησιμοποιούν το Web Audio API

Beats Audio API – demo (*https://jmperezperez.com/beats-audio-api*)

Αυτό το demo χρησιμοποιεί το Web Audio API για να προσδιορίσει την ταχύτητα (tempo) ενός τραγουδιού, επεξεργαζόμενο τα δεδομένα από ένα απόκομμα 30 δευτερολέπτων. Σε πρώτη φάση το σήμα κατευθύνετε μέσα από ένα χαμηλοπερατό φίλτρο και στη συνέχεια από την έξοδο του φίλτρου επιλέγονται οι κορυφές. Τον τελικό υπολογισμό τον κάνει από τα ομαδοποιημένα διαστήματα μεταξύ των κορυφών. Γενικώς αρκετά απλή εφαρμογή με όχι και τόσο ασφαλή αποτελέσματα κυρίως για ηχητικά κομμάτια που δεν περιλαμβάνουν κρουστά.

Pitchdetector - demo (*https://webaudiodemos.appspot.com/pitchdetect/index.html*)

Η συγκεκριμένη εφαρμογή πειραματίζεται πάνω στην αναγνώριση τονικού ύψους. Χρησιμοποιεί το Web Audio API και προβάλει τα αποτελέσματα σε πραγματικό χρόνο απεικονίζοντας το σύμβολο της τελευταίας νότας που παίχτηκε (π.χ. D). Χρησιμοποιεί το ηχητικό σήμα της εξόδου μιας διεπαφής AnalyserNode από όπου περνάει το αρχικό σήμα, για να υλοποιήσει τον αλγόριθμο αυτοσυσχέτησης. Έχει καλά αποτελέσματα σε ένα σφύριγμα, όμως έχει μερικά σφάλματα σε εκτίμηση τονικού ύψους όταν ο ήχος προέρχεται από όργανα με αντηχείο.

Web Audio Playground – demo (*http://webaudioplayground.appspot.com*)

Απλή εφαρμογή φτιαγμένη για λόγους εκμάθησης και επίδειξης των AudioNodes και της σπονδυλωτής δρομολόγησης του Web Audio API.

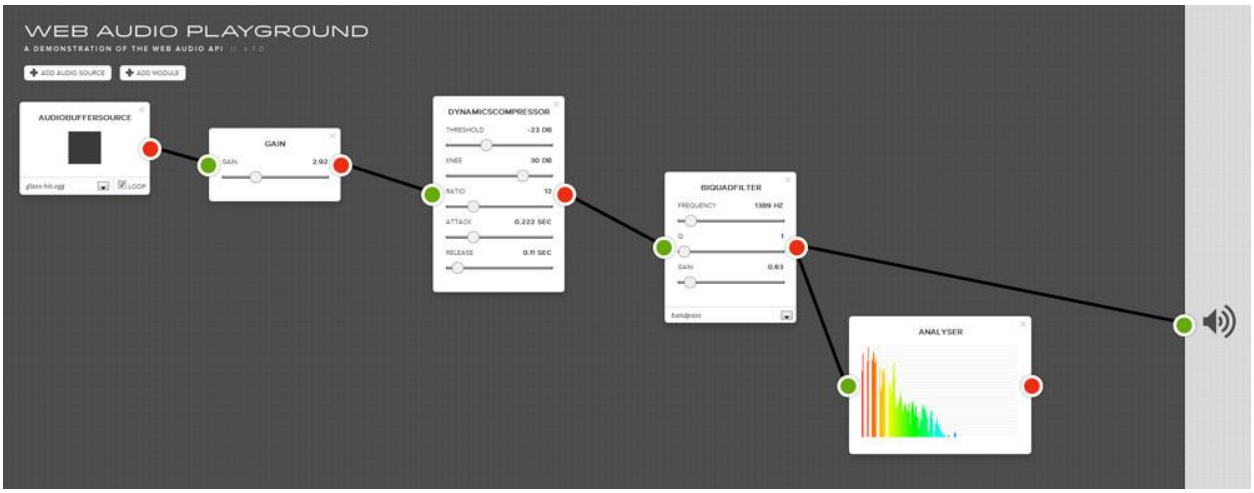


Figure 3.32 Web Audio Playground

GraphicalFilterEditor - demo (*http://carlosrafaelgn.com.br/GraphicalFilterEditor/index.html*)

Αυτή είναι μία παλιά εφαρμογή γραμμένη σε C++ ή οποία έγινε port σε javascript χρησιμοποιώντας HTML5, Web Audio API, Web Worker API και File API. Ο χρήστης μπορεί να επεξεργαστεί με γραφικό τρόπο ένα φίλτρο εξισωτή (*equalizer filter*) ο οποίος επηρεάζει το ηχητικό σήμα της εισόδου σε πραγματικό χρόνο. Επίσης προσφέρει τη δυνατότητα φιλτραρίσματος και ενός αρχείου ήχου ασύγχρονα.

3.2.2 Εφαρμογές απεικόνισης μουσικής πληροφορίας

Pianoroll-JS (*http://www.oliphaunts.com/pianoroll-js*)

Το Pianoroll.js είναι μία διαδικτυακή εφαρμογή η οποία χρησιμοποιεί μια βιβλιοθήκη που ονομάζεται audiolet. Το οπτικό κομμάτι είναι πλήρως γραμμένο σε HTML5 canvas. Η εφαρμογή αυτή είναι ένας απλός μουσικός sequencer ο οποίος επιτρέπει βασική

εισαγωγή και αφαίρεση νοτών, μερικά απλά συνθετικά όργανα και αναπαραγωγή. Επίσης για την παραγωγή των ήχων χρησιμοποιείται το Web Audio API.

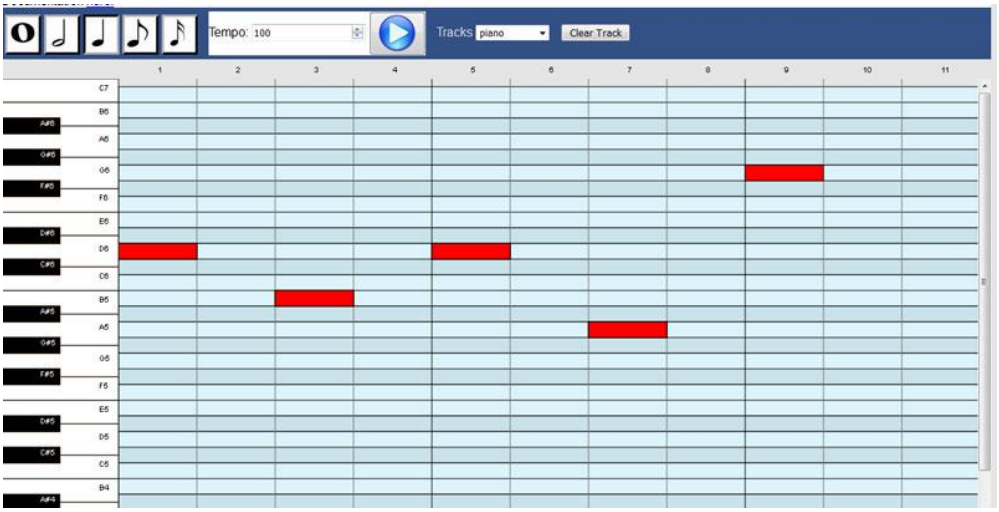


Figure 3.33 Το γραφικό περιβάλλον MIDI της εφαρμογής Pianoroll-js

My VexFlow (*http://my.vexflow.com*)

Το My VexFlow επιτρέπει στους χρήστες του να δημοσιεύσουν περιεχόμενο με καποια μουσική σημειογραφία, ταμπλατούρα κιθάρας καθώς και τα διαγράμματα συγχορδιών. Αυτά όλα επιτυγχάνονται με το VexFlow API. Το VexFlow είναι ένα API αναπαράστασης μουσικής σημειογραφίας. Είναι γραμμένο σε javascript και υποστηρίζει HTML5 Canvas και SVG.

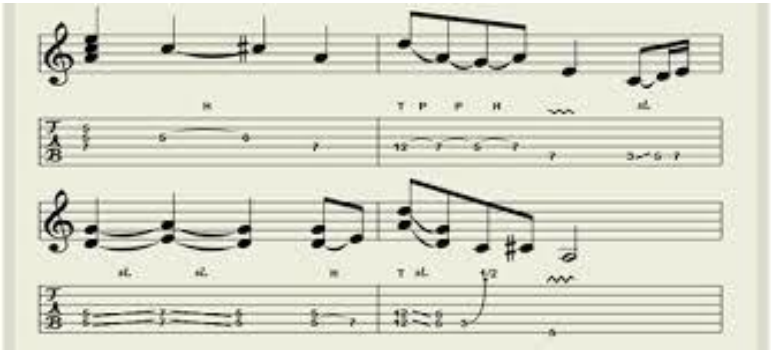


Figure 3.34 Παρτιτούρα του Vexflow API

Soundslice (*https://www.soundslice.com/scores/auld-lang-syne*)

Το Soundslice είναι μία ιστοσελίδα που βοηθά μουσικούς να μαθαίνουν τραγούδια. Είναι ένα σύστημα αναπαραγωγής για ταμπλαδούρες και παρτιτούρες το οποίο αναπαράγει το περιεχόμενο τους ηχητικά ενώ το δείχνει οπτικά. Συγχρονίζει τη μουσική σημειογραφία με ήχο. Χρησιμοποιεί το Web Audio API για διεργασίες όπως αλλαγή τονικού ύψους, αναπαραγωγή δειγμάτων ήχου κλπ.

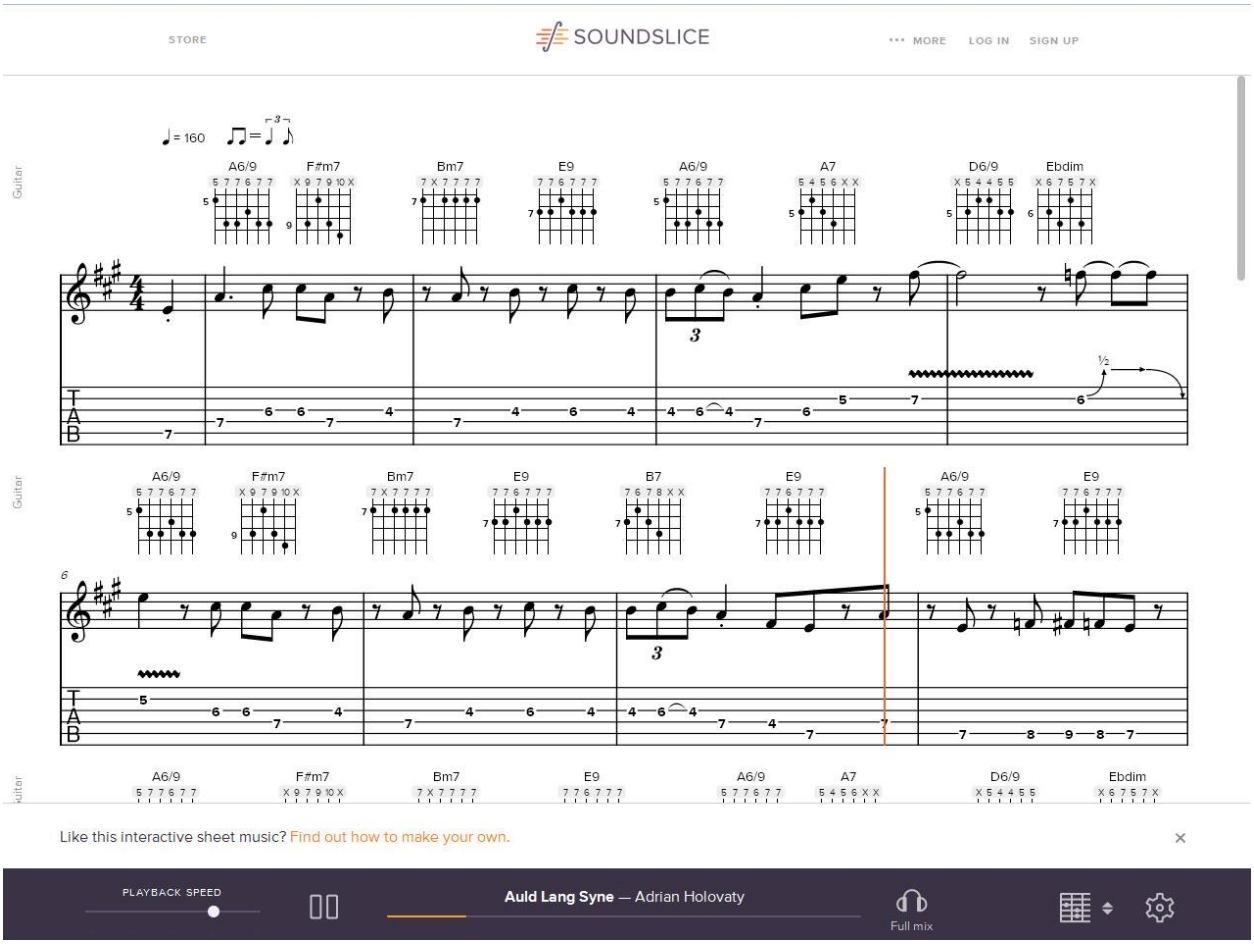


Figure 3.35 Περιβάλλον Soundslice

4 Note Recognizer

4.1 Απαιτήσεις χρηστών από μία εφαρμογή ανάκτησης μουσικής πληροφορίας από ηχητικά σήματα

Η θεματική γύρω από την οποία κινείται η εφαρμογή έχει αρκετά μεγάλη έκταση και στην ουσία ένας τελικός στόχος γι αυτή τη θα μπορούσε να οριστεί η κατασκευή ενός συστήματος που θα ακούει και θα καταλαβαίνει ότι και ένας εκπαιδευμένος μουσικός. Η ιδέα ξεκινάει από μία απλή ανάγκη ενός μουσικού σήμερα. Είναι η ανάγκη της καταγραφής μίας οργανικής απόδοσης. Ποιο αναλυτικά είναι βοηθητικό για οποιονδήποτε μουσικό να αναπαράγει ένα μουσικό κομμάτι μπροστά στο μικρόφωνο και να βλέπει στην οθόνη την ηχητική πληροφορία μεταφρασμένη σε μουσική σημειογραφία. Το να αναπαράγει ένα ολόκληρο κομμάτι και να δει στο τέλος τι έπαιξε είναι πολύ σημαντικό, τον αποτρέπει από μία διαδικασία απομνημόνευσης, κατά τη διάρκεια της αναπαραγωγής, η οποία θα είχε αντίκτυπο και στην απόδοση. Εξίσου σημαντικό όμως είναι να είναι δυνατή η απεικόνιση της μουσικής πληροφορίας και σε πραγματικό χρόνο. Ας υποθέσουμε ότι ένας μουσικός/δάσκαλος ωδείου παίζει ένα επαναλαμβανόμενο μοτίβο το οποίο καταγράφεται στο σύστημα. Αν η απεικόνιση της μουσικής σημειογραφίας γινόταν στο τέλος της αναπαραγωγής κατ απαίτηση, δεν θα ήταν δυνατόν για ένα δεύτερο μουσικό/μαθητή να συμμετάσχει στην αναπαραγωγή πριν αυτή ολοκληρωθεί, με την υπόθεση ότι δεν ξέρει τις νότες και λόγο εμπειρίας δεν μπορεί να τις αντιληφθεί ακουστικά. Επίσης ας υποθέσουμε ότι αυτοί οι δύο μουσικοί ήταν σε κάποιο εξωτερικό χώρο χωρίς υπολογιστή. Αυτή η συνθήκη κάνει την διαθεσιμότητα μίας τέτοια εφαρμογής και σε κινητά πολύ χρήσιμη. Απαραίτητα σε ένα τέτοιο λογισμικό δεν παύουν να είναι και πιο κλασσικά εργαλεία. Εργαλεία όπως το κουρδιστήρι και η ένδειξη ταχύτητας αναπαραγωγής. Έτσι συμπεραίνουμε ότι οι παρούσες τεχνολογικές συνθήκες προσφέρονται για την υλοποίηση ενός λογισμικού που θα προσφέρει αναγνώριση τονικότητας, ταχύτητας και onset ενός ηχητικού αποσπάσματος, θα απεικονίζει την πληροφορία σε πραγματικό χρόνο, θα είναι εύκολα προσβάσιμο από οποιονδήποτε, βάση της διαλειτουργικότητας του σε οποιαδήποτε συσκευή, και από οπουδήποτε, μέσω του διαδικτύου. Σαφώς τα όρια σε επίπεδο λειτουργικοτήτων βάση των αναγκών των χρηστών δεν είναι αυτά, είναι πολύ περισσότερα, από την δυνατότητα αναπαραγωγής, επεξεργασίας και εξαγωγής της μουσικής πληροφορίας μέχρι την απομακρυσμένη ανταλλαγή μουσικής πληροφορίας της αναπαραγωγή ενός μουσικού κομματιού σε πραγματικό χρόνο. Η παρούσα έκδοση της εφαρμογή θα περιλαμβάνει λειτουργικότητες που αφορούν την βασική υλοποίηση ενός τέτοιου συστήματος με στόχο την σταθερότητα και την προσωρινά αποτελεσματική χρηστικότητα του από κάποιο χρήστη.

4.2 Προδιαγραφές συστήματος

Για να επιτευχθεί η ανάπτυξη μίας τέτοιας εφαρμογής θα πρέπει να γίνει μία εκτίμηση των κομματιών που θα αποτελείτε, μία σύνδεση μεταξύ τους και μία κατάταξη τους σε λογική σειρά. Βάση των αναγκών και της έρευνας το σύστημα πρέπει να αποτελείτε από τα εξής κομμάτια/μηχανισμούς με την ακόλουθη λογική σειρά:

1. **Βασικές λειτουργίες περιβάλλοντος ήχου σε πλαίσιο τεχνολογιών διαδικτύου**

   Ελάχιστες προδιαγραφές: Ανάπτυξη συστήματος για την αναπαραγωγή και επεξεργασία μουσικής μέσω βασικών μοντέλων των φυλλομετρητών.

   Βέλτιστες προδιαγραφές: Υλοποίηση περιβάλλοντος ήχου που βασίζεται σε ένα ευρέως διαδεδομένο API με μεγάλη κοινότητα και ήδη ανεπτυγμένα μοντέλα για αναπαραγωγή και επεξεργασία ήχου.

2. **Αναγνώριση onsets ηχητικού αποσπάσματος**

   Ελάχιστες προδιαγραφές: Μηχανισμός που θα εντοπίζει τα onset στο τέλος της αναπαραγωγής και θα τα ταξινομεί στα αποτελέσματα.

   Βέλτιστες προδιαγραφές: Δυνατότητα ακριβούς εκτίμησης σε πραγματικό χρόνο με υλοποίηση τεχνικών βελτίωσης επιδόσεων για την εκτέλεση των αλγορίθμων που απατούνται.

3. **Αναγνώριση τονικότητας ηχητικού αποσπάσματος**

   Ελάχιστες προδιαγραφές: Αναγνώριση τονικού ύψους με την παραδοχή ότι τα όργανα με αντήχηση θα παρουσιάζουν μη ακριβή αποτελέσματα.

   Βέλτιστες προδιαγραφές: Η αναγνώριση του τονικού ύψους θα εφαρμόζει τους κατάλληλους αλγορίθμους ώστε να ελαχιστοποιεί τα μη επιθυμητά χαρακτηριστικά του ηχητικού σήματος που έχουν να κάνουν με τις ιδιότητες αντήχησης του υλικού και με τον εξωτερικό θόρυβο, καθώς και θα υλοποιεί αλγορίθμους εκτίμησης ορθότητας και διόρθωσης αποτελέσματος σε πραγματικό χρόνο.

4. **Υπολογισμός tempo ηχητικού αποσπάσματος**

   Ελάχιστες προδιαγραφές: Με βάση την προηγούμενη αναγνώριση των onset, της οποίας αυτός ο υπολογισμός είναι παράγωγο, μπορεί να υπάρξει η παραδοχή της ύπαρξης ενός κρουστικού ήχου στο ηχητικό σήμα για ακριβές αποτέλεσμα.

Βέλτιστες προδιαγραφές: Υλοποίηση βέλτιστων προδιαγραφών στην αναγνώριση των onset ώστε ο υπολογισμός της ταχύτητας να είναι ακριβείς. Επιπλέον σύγκριση αποτελεσμάτων σε περιπτώσεις εναλλαγών ταχύτητας και επικράτηση της συχνότερης.

5. **Χαρακτηρισμός νοτών**

   Ελάχιστες προδιαγραφές: Οι νότες δεν χαρακτηρίζονται, απλά απεικονίζονται απευθείας.

   Βέλτιστες προδιαγραφές: Υπάρχει χαρακτηρισμός που συνθέτεται από τα στιγμιότυπα του συστήματος και τη μουσική πληροφορία που έχει να κάνει με το τονικό ύψος, τη διάρκεια, την αρχή και το τέλος της κάθε νότας. Αυτός ο χαρακτηρισμός οδηγεί στην δημιουργία αντικειμένων με συγκεκριμένα χαρακτηριστικά.

6. **Απεικόνιση νοτών**

   Ελάχιστες προδιαγραφές: Οι νότες απλά απεικονίζονται σε ένα περιβάλλον χωρίς διάδραση από το χρήστη.

   Βέλτιστες προδιαγραφές: Υπάρχει δυνατότητα ο χρήστης να μετακινήσει και να επεξεργαστεί τη διάρκεια των απεικονισμένων αντικειμένων πάνω στο περιβάλλον αναπαράστασης.

7. **Μηχανισμός διαλειτουργικότητας συσκευών**

   Ελάχιστες προδιαγραφές: Προσφέρεται η δυνατότητα χρήσης της εφαρμογής με όλα της τα χαρακτηριστικά χωρίς όμως να προσαρμόζεται οπτικά σε διαφορετικού μεγέθους οθόνης συσκευές.

   Βέλτιστες προδιαγραφές: Η εφαρμογή είναι διαθέσιμη σε χρήστες όλων των συσκευών με όλα της τα χαρακτηριστικά και προσαρμοσμένο γραφικό περιβάλλον ανάλογα με τις ιδιότητες της κάθε συσκευής.

Οι ελάχιστες προδιαγραφές παραπάνω αντικατοπτρίζουν και τους ελάχιστους βασικούς στόχους για την ολοκλήρωση της πτυχιακής εργασίας.

4.3 Αλγόριθμοι και Μοντέλα που θα χρησιμοποιηθούν για την υλοποίηση

4.3.1 Web Audio API

Η κύρια βάση γι' αυτή την εφαρμογή θα είναι το Web Audio API. Οι λόγοι που επιλέχθηκε είναι:

- Διαθέτει υλοποιημένους κόμβους οι οποίοι έχουν να κάνουν με την επεξεργασία σήματος(DynamicCompressorNode, BiquadFilterNode, AnalyserNode, GainNode κ.α.)
- Είναι η τελευταία λέξη της τεχνολογίας στο χώρο του διαδικτιακού ήχου και έχει μεγάλα περιθώρια ανάπτυξης.
- Η ευρεία χρήση του σε συνδυασμό με το ότι είναι διεπαφή ανοιχτού κώδικα κάνει την κοινότητα υποστήριξης μεγαλύτερη.

4.3.2 Εκτίμηση της αρχής της νότας (*Onset detection*) & εύρεση ρυθμικής πληροφορίας (*BPM calculation*)

Ένα βασικό υποσύστημα της εφαρμογής είναι ο προσδιορισμός των onsets και ο υπολογισμός του tempo. Για να αντλήσουμε αποτέλεσμα για την κάθε περίπτωση θα πρέπει να ακολουθήσουμε μία ακολουθία βημάτων, τα οποία δεν είναι προκαθορισμένα. Υπάρχουν πολλές μέθοδοι για να φθάσουμε σε ένα τελικό αποτέλεσμα. Στην περίπτωση της παρούσας εφαρμογής το υποσύστημα αυτό αποτελείτε απο:

- **Dynamic Compressor (*Web audio API*)**

  Γενικά ένας δυνατότερος και πιο πλούσιος ήχος βελτιώνει την απόδοση των αλγορίθμων που θα εντοπίσουν τα onsets στη συνέχεια.

- **AnalyserNode (*Web audio API*)**

  Αν χρησιμοποιούνταν το ακατέργαστο ηχητικό σήμα (PCM) απευθείας θα είχαμε τεράστιες απώλειες απόδοσης. Έτσι ενδιάμεσα προστέθηκε ένας AnalyserNode ώστε η έκταση των δεδομένων στην έξοδο να είναι μικρότερα.

- Κατά τη διάρκεια της έρευνας αλγορίθμων onset detection υλοποιημένους σε javascript βρέθηκε μόνο μία (θεωρητική υλοποίηση) η οποία και δοκιμάστηκε. [21] Αυτή η θεωρητική υλοποίηση περιλαμβάνει τις μεθόδους **STFT** και **HFC / SD / PD** που περιγράφονται στα κεφάλεα 4.1.4.1 και 4.1.4.2.
  Τα παραπάνω ελέγχθηκαν σε θεωρητικό επίπεδο και υλοποιούν θεωρητικές προσεγγίσεις στο θέμα. Σημαντικό κριτήριο αξιολόγησης σχετικά με την επιλογή τους ήταν: το κέρδος σε χρόνο, αφού δεν θα χρειαστεί η υλοποίηση τους, η έλλειψη εμπεριστατωμένης γνώσης σχετικά με την επεξεργασία σήματος, ώστε να γίνει παραγωγή πρωτότυπου κώδικά και η ποιότητα των αποτελεσμάτων.

- Η **κανονικοποίηση (*Normalization*)** του σήματος, αν και στην προαναφερθείσα θεωρητική υλοποίηση αναφέρετε ως κομμάτι του STFT, στην παρούσα εφαρμογή χρησιμοποιήθηκε μετά τις μεθόδους ανίχνευσης. Δοκιμάστηκαν πολλές εκδοχές αλλά η ποιότητα των αποτελεσμάτων ήταν καλύτερη σε αυτή τη θέση. Η συνάρτηση εκτελείται μέσα σε ένα Web Worker διότι καλείτε να υπολογίσει πολλές φορές το δευτερόλεπτο κανικοποιημένες τιμές για πίνακες που έχουν όλο και μεγαλύτερο μέγεθος.

- Το **φίλτρο ενδιάμεσης τιμής (*Median filter*)** είναι υλοποίηση από το https://github.com/mikolalysenko/moving-median, προσαρμόστηκε ώστε να τρέχει στη μεριά του πελάτη (client side). Η μέθοδος επιλογής κορυφών που υλοποιείτε είναι πρωτότυπος κώδικας. Και οι δύο υλοποιήσεις βασίζονται στη θεωρητική προσέγγιση που περιγράφεται στο κεφάλαιο 4.1.4.3.

- Ο υπολογισμός των **χτύπων ανά λεπτό (*BPM*)** έγινε με βάση τη θεωρία που περιγράφεται στο κεφάλαιο 4.1.5. Γι αυτό το σημεία χρησιμοποιήθηκε μέρος του κώδικα από την εφαρμογή Beats Audio API που περιγράφεται στο κεφάλαιο 3.1.

4.3.3 Εκτίμηση του τονικού ύψους της νότας (*Pitch detection*)

Το βασικότερο υποσύστημα της εφαρμογής. Το Web Audio API ως περιβάλλον ήχου βοήθησε πολύ σε σχέση με αυτό το υποσύστημα. Τα κομμάτια τα οποία το απαρτίζουν είναι ένας παραμετρικός ισοσταθμιστής (*Parametric EQ*), ένας analyserNode και μια υλοποίηση του αλγορίθμου της αυτοσυσχέτησης.

- Ο **παραμετρικός ισοσταθμιστής (*Parametric EQ*)** εισήχθη με βάση τον κανόνα του εύρους συχνοτήτων ανά μέσω αναπαραγωγής. Με ποιο απλά λόγια το κάθε μουσικό όργανο έχει ένα εύρος συχνοτήτων όταν αναπαράγει μία νότα, οπότε το σύστημα κρατάει της συχνότητες αυτού του εύρους και κόβει τις υπόλοιπες, έτσι ελαχιστοποιεί ένα μέρος του θορύβου και βελτιώνει το τελικό αποτέλεσμα. Στην

παρούσα εφαρμογή ο παραμετρικός ισοσταθμιστής αποτελεί ένα σύνολο BiquadFilterNodes με ανάλογες ρυθμίσεις στο κάθε ένα και μία σειρά από προεπιλογές συμβατές με κάθε τύπο οργάνου. Η βασική υλοποίηση του Parametric EQ είναι κομμάτι της εφαρμογής https://github.com/ericschmidt/noise-reduction , με κάποιες προσαρμογές ώστε να λειτουργεί αυτόνομα, διαφορετικές ρυθμίσεις κα προεπιλογές.

- Στην αναγνώριση του τονικού ύψους μεγάλο ρόλο για την εκτίμηση των χαμηλών συχνοτήτων παίζει το μέγεθος του παραθύρου fft. Ο **analyserNode**, ο οποίος και χρησιμοποιείτε, έχει μέγιστο περιορισμό μεγέθους του fft παραθύρου που εξάγει στα 2048. Το μέγεθος του παραθύρου σε συνδυασμό με το ρυθμό δειγματοληψίας ,παίζει καθοριστικό ρόλο στην ανάλυση συχνότητας που θα έχει ο αλγόριθμος αναγνώρισης , όσο μικρότερο το παράθυρο τόσο υψηλότερη η τελευταία νότα χαμηλών συχνοτήτων που θα μπορεί να αναγνωρίσει με ασφάλεια, το οποίο συμβαίνει λόγο της λογαριθμικότητας των νοτών. Από την άλλη ο scriptProcessorNode έχει πολύ μεγαλύτερα όρια στο μήκος του παραθύρου (16384 samples), όμως δεν υλοποιεί fft μετασχηματισμό και έχει εγκαταλειφθεί από το Web Audio API. Ο AudioWorkerNode που είναι ο αντικαταστάτης του, σε αυτό το θέμα έχει και αυτός τις ίδιες ιδιότητες.

- Η εφαρμογή χρησιμοποιεί τον **αλγόριθμο αυτοσυσχέτησης (*Autocorrelation*)** για να κάνει εκτίμηση του τονικού ύψους, ο οποίος περιγράφηκε στο κεφάλαιο 4.1.6.2,. Το μεγαλύτερο μέρος αυτής της υλοποιήσεις έγινε με τη βοήθεια του κώδικα από το Pitchdetector που παρουσιάζεται στο κεφάλαιο 3.1. Ο αλγόριθμος αυτοσυσχέτησης διαχωρίστηκε σε ένα Web Worker λόγο των συνεχών υπολογισμών, ώστε να μην υπάρξει επίπτωση στις επιδόσεις. Τα αποτελέσματα αυτής της υλοποίησης είναι αρκετά ικανοποιητικά παρά τις περιορισμένες δυνατότητες (αναγνώριση στις χαμηλές συχνότητες), λόγο του analyserNode. Ο συνδυασμός αυτού και της επίπτωσης στις επιδόσεις που θα προέκυπτε αν το μέγεθος παραθύρου ήταν, για παράδειγμα 16384 (scriptProcessorNode ή AudioWorkerNode), μας οδηγεί στην επιλογή του analyserNode.

4.3.4 Διαλειτουργικότητα

Η παρούσα εφαρμογή βασίζεται σε τεχνολογίες HTML5 και javascript οι οποίες είναι συμβατές με όλως των ειδών τις συσκευές (desktop, tablet, mobile), έτσι το μόνο που απομένει για τη λειτουργία του συστήματος σε αυτές είναι η υλοποίηση responsive

design, το οποίο βάση κάποιων κανόνων προσαρμόζει τη διεπαφή στο μέγεθος της οθόνης του χρήστη.

4.4 Αλγόριθμοι και μοντέλα που δημιουργήθηκαν για τις απαιτήσεις της παρούσας εφαρμογής

Βάση των απαιτήσεων της εφαρμογής υπήρξε η ανάγκη προσέγγισης κάποιων θεμάτων σε βάθος. Οι δύο βασικότερες απαιτήσεις που τίθενται είναι η καταγραφή και αναπαράσταση των αποτελεσμάτων στην πάροδο του χρόνου και η δυνατότητα άντλησης των αποτελεσμάτων από τους αλγορίθμους σε πραγματικό χρόνο.

4.4.1 Νήμα εκτέλεσης (*thread*) και στιγμιότυπα (*snapshots*)

Για να επιτευχθεί η άντληση αποτελεσμάτων από τους αλγορίθμους σε πραγματικό χρόνο θα πρέπει να υπάρχει μία συνεχής ροή του εισερχόμενου σήματος προς τους αλγόριθμους αυτούς και μία συνεχής καταγραφή της μουσικής πληροφορίας από τους αλγορίθμους στο σύστημα. Για να επιτευχθεί αυτό, η εφαρμογή χρησιμοποιεί ένα **νήμα εκτέλεσης (thread)** ως βασικό πυλώνα της. Αυτή η λειτουργία στην ουσία στέλνει και λαμβάνει πληροφορία από τις υπορουτίνες συνεχώς. Το νήμα εκτέλεσης λειτουργεί συνεχώς αφού εκτελείτε πεπερασμένες φορές ανά δευτερόλεπτο. Κάθε φορά που εκτελείτε την ονομάζουμε **κύκλο (cycle)** και το αντικείμενο που ανήκει η κάθε πληροφορία που το σύστημα αντλεί το ονομάζουμε **στιγμιότυπο (snapshot).** Από κάθε κύκλο μπορεί να παραχθεί ένα μοναδικό στιγμιότυπο του οποίου **η χρονική θέση ισούται με την απόκλιση του χρονικού στίγματος του τρέχοντος κύκλου από το χρονικό στίγμα του πρώτου κύκλου**.
Το ίδιο νήμα εκτέλεσης χρησιμοποιείται και για την απεικόνιση της πληροφορίας, η οποία στην ουσία είναι η αναπαράσταση της μουσικής πληροφορίας του στιγμιότυπου του τρέχοντος κύκλου.

4.4.2 Χρονική κατανομή εκτιμήσεων μουσικής πληροφορίας

Η μουσική και η απεικόνιση της είναι άμεσα συναρτώμενες με το χρόνο. Όταν περιγράφεται μια σύνθεση σε μουσική σημειολογία πάντα ο άξονας x οποιουδήποτε

κοινού τρόπου απεικόνισης είναι ο χρόνος. Η παρούσα εφαρμογή χρησιμοποιεί ως βασικό τρόπο απεικόνισης της πληροφορίας το περιβάλλον αναπαράστασης MIDI, στο οποίο λόγο της απλότητας και της υβριδικότητας του βασίστηκε η λογική των αλγορίθμων που αναπτύχθηκαν. Οπότε **ο άξονας x της παρούσας εφαρμογής αποτελείτε από απεικονίσεις της πληροφορίας του κάθε στιγμιότυπου**.
Όσο το νήμα εκτέλεσης εκτελείται οι αλγόριθμοι εκτιμούν αν το τρέχον στιγμιότυπο είναι αρχή μίας νότας (*onset*) καθώς το τονικό του ύψος, αν υπάρχει. Σε κάθε κύκλο όλες αυτές οι πληροφορίες ωθούνται σε ένα κεντρικό πίνακα στιγμιότυπων, του οποίου οι θέσεις είναι άμεσα συναρτώμενες με τους κύκλους άρα και με το χρόνο. Ο πίνακας στιγμιότυπων χρησιμοποιείτε για την επεξεργασία και την αναπαράσταση της πληροφορίας.

4.4.3 Σχετικότητα χώρου και χρόνου

Μπορούμε να πούμε ότι όπως ένα ηχητικό σήμα εκφράζεται ψηφιακά σε ένα πίνακα τιμών έτσι και η μουσική πληροφορία αυτού του σήματος εκφράζεται από τον πίνακα στιγμιότυπων του. Αυτοί οι δύο πίνακες όμως δεν είναι κοινής έκτασης. Αυτό συμβαίνει διότι το μέγεθος και των δύο, δηλαδή το πλήθος των τιμών τους, προκύπτει από τον ρυθμό δειγματοληψίας πολλαπλασιασμένο με τον χρόνο του σήματος. **Ο ρυθμός δειγματοληψίας του πίνακα στιγμιότυπων είναι ίσος με τον αριθμό των κύκλων ανά δευτερόλεπτο**. **Στην παρούσα εφαρμογή ο ρυθμός δειγματοληψίας του πίνακα στιγμιότυπων είναι περίπου 50Hz**. Για παράδειγμα αν έχουμε ένα ηχητικό σήμα 10 δευτερολέπτων το πλήθος των τιμών του πίνακα του σήματος θα είναι 44100 x 10 = 441000 [1], ενώ του πίνακα στιγμιότυπων θα είναι 50 x 10 = 500 τιμές. Η απεικόνιση της πληροφορίας σε πραγματικό χρόνο που πρέπει να γίνεται κάνει την αναπαράσταση του άξονα χρόνου x στο χώρο λίγο πολύπλοκη. Θα ήταν πολύ απλό αν η απεικόνιση των μουσικολογικών χαρακτηριστικών γινόταν κατ απαίτηση γιατί θα είχαμε το σύνολο της έκτασης του σήματος και θα μπορούσαμε εύκολα να υπολογίζουμε τη σχετικότητα χώρου και χρόνου. Αν για το παραπάνω παράδειγμα ο χώρος απεικόνισης ήταν 1000px θα έπρεπε η έκταση του κάθε δευτερολέπτου να είναι 1000px/10sec = 100px το οποίο σημαίνει ότι κάθε στιγμιότυπο μπορεί να αναπαρασταθεί 100px / 50cycles/sec = 2px.
Τα δευτερόλεπτα δεν είναι η μόνη μονάδα χρόνου που θα χρησιμοποιηθεί στην εφαρμογή, θα χρησιμοποιηθούν επίσης οι μουσικές μονάδες διάρκειας χτύπος (*beat*) και τα μέτρα (*bars*) των οποίων η χρονική διάρκεια, οπότε και έκταση στον άξονα x, καθορίζεται από την ταχύτητα αναπαραγωγής (*tempo*) των χτύπων ανά λεπτό (*BPM*).

[1] Υποθέτουμε ότι το σήμα έχει το συνηθισμένο ρυθμό δειγματοληψίας των 44100 Hz.

Στο παράδειγμα μας αν το tempo ήταν 120 BPM τότε θα έπρεπε η έκταση του κάθε beat να είναι:

120BPM / 60sec = 2 BPS[2]
100px / 2BPS = 50px

Όποτε και το μέγεθος ενός μέτρου θα είναι 50px x 4beat = 200px.
Το μήκος/έκταση των νοτών καθορίζεται από τη μουσική μονάδα διάρκειας beat οπότε ο υπολογισμός της έκτασης της στο χώρο γίνεται κατά τον ανωτέρω τρόπο. Έτσι αντιλαμβανόμαστε πως όσο ποιο γρήγορα αναπαράγεται ένα μουσικό κομμάτι τόσο μικραίνει η έκταση των νοτών που απεικονίζονται.

4.4.4 Υπολογισμός σχετικότητας μεγεθών σε πραγματικό χρόνο

Όταν η απεικόνιση της ηχητικής πληροφορίας πρέπει να γίνει σε πραγματικό χρόνο δεν ξέρουμε τη διάρκεια του ηχητικού αποσπάσματος. Επίσης γνωρίζουμε ότι ο αλγόριθμος υπολογισμού της ταχύτητας αναπαραγωγής κάνει κάποιο αρχικό χρόνο μέχρι να παρουσιάσει το πρώτο αποτέλεσμα. Οι παραπάνω δύο παράμετροι, περιγράφηκε στο προηγούμενο κεφάλαιο πόσο απαραίτητοι είναι για τον καθορισμό των μεγεθών στο χώρο.

Για να ξεπεραστεί αυτό το πρόβλημα η εφαρμογή χρησιμοποιεί δύο αρχικές παραδοχές. Η πρώτη είναι ότι το πεδίο αναπαράστασης MIDI (*pianoroll*) έχει ένα πεπερασμένο μέγεθος και η δεύτερη ότι η εφαρμογή ξεκινά με ένα κοινά αποδεκτό μέσο BPM της τάξεως των 120 μέχρι ο αλγόριθμος υπολογισμού ταχύτητας να επιστρέψει το πρώτο αποτέλεσμα. Έτσι τα μεγέθη των μέτρων σχεδιάζονται κατά την εκκίνηση της εφαρμογής βάση των υπολογισμών που βασίζονται στις αρχικές τιμές και στη συνέχεια κατ απαίτηση του χρήστη μπορεί να γίνει επανασχεδιασμός βάση των νέων μεγεθών που υπολογίζονται βάση της πραγματικής ταχύτητας του μουσικού αποσπάσματος.

[2] Όπου BPS χτύποι ανά δευτερόλεπτο

4.4.5 Διορθώσεις εκτιμήσεων τονικού ύψους με βάση τη χρονική κατανομή

Μέσα από τις συνεχείς δοκιμές παρατηρήθηκε μία απώλεια ακρίβειας αποτελέσματος που προέρχεται από κάποιες εσφαλμένες εκτιμήσεις που επιστρέφει ο αλγόριθμος αυτοσυσχέτησης. Σφάλματα που συνήθως έχουν να κάνουν με θόρυβο και αντήχηση. Αφού η βελτίωση του ίδιου του αλγορίθμου δεν ήταν εφικτή λόγο έλλειψης προχωρημένων θεωρητικών γνώσεων στην επεξεργασία σημάτων αποφασίστηκε να χρησιμοποιηθούν κάποιες λογικές παραδοχές για να γίνεται μία υποτυπώδης διόρθωση του αποτελέσματος σε πραγματικό χρόνο.
Στην ουσία το βασικό εργαλείο σε αυτή την περίπτωση είναι ο πίνακας στιγμιότυπων μέσω του οποίου έχουμε τη δυνατότητα να κάνουμε συγκρίσεις μεταξύ των στιγμιότυπων που βασίζονται σε κάποιους βασικούς κανόνες. Έτσι σε κάθε κύκλο το νήμα εκτέλεσης καλεί μία συνάρτηση η οποία εξετάζει τη μουσική πληροφορία του στιγμιότυπου της προπροηγούμενης νότας και των δύο επόμενων και προηγούμενων αυτής. Κάπου εδώ πρέπει να αναφερθεί και ένας περιορισμός του αλγορίθμου αυτού. Λόγο του ρυθμού δειγματοληψίας του πίνακα στιγμιότυπων ο οποίος είναι 50Hz συμπεραίνουμε ότι οι ασφαλείς διορθώσεις αποτελεσμάτων που μπορούμε να έχουμε μπορούν να καλύψουν μέχρι 1/16 διάρκεια νότας και όχι μικρότερη. Αυτό συμβαίνει γιατί για μικρότερη διάρκεια νότας έχουμε πολύ περιορισμένο αριθμό στιγμιότυπων και δεν μπορούμε να ακολουθήσουμε ασφαλώς τους παρακάτω κανόνες (στα 120 BPM τα στιγμιότυπα που εκφράζουν μία νότα 1/16 είναι περίπου 12).

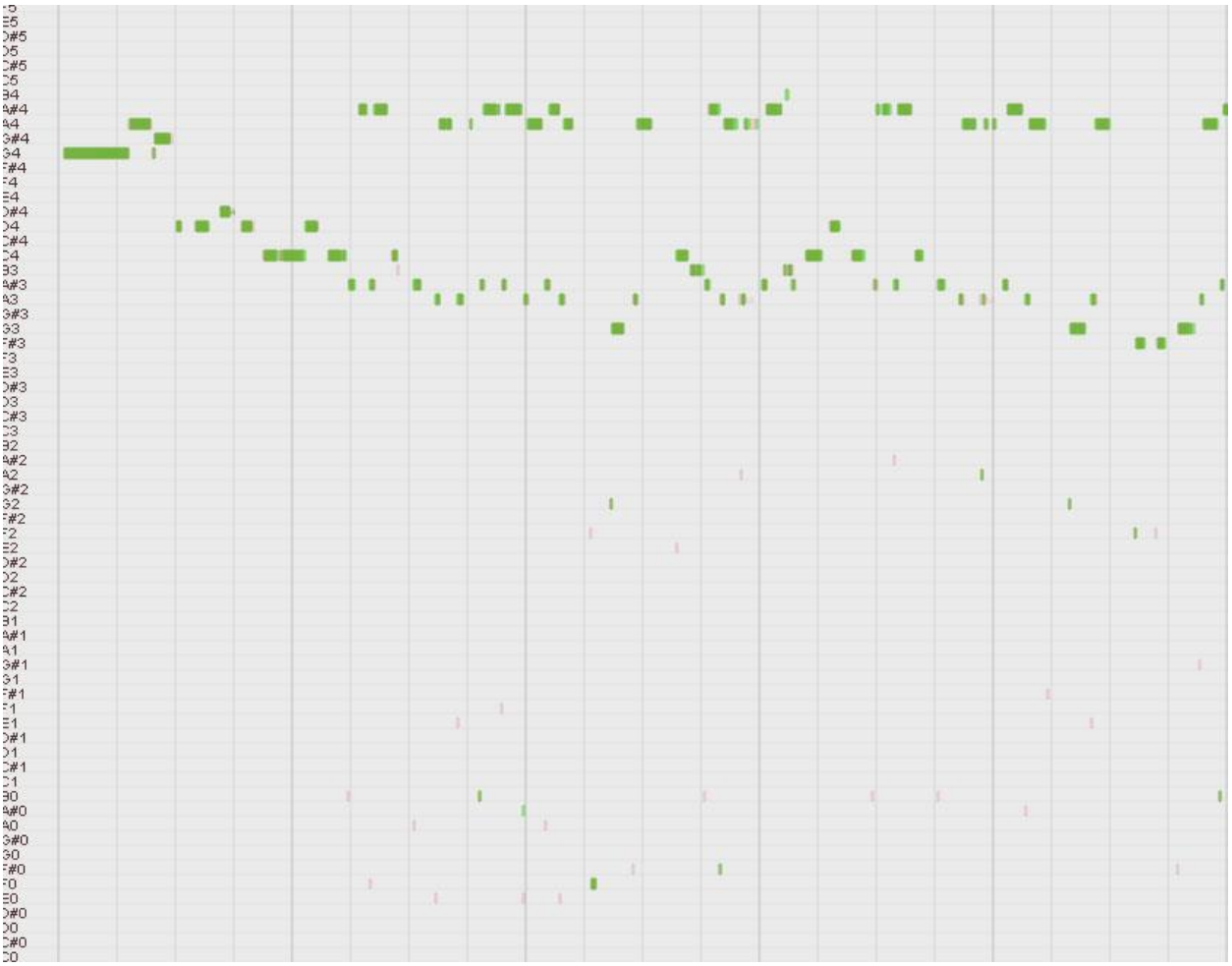
Οι βασικές λογικές παραδοχές που ακολουθεί αυτός ο αλγόριθμος είναι:

1. Αφού η εκτίμηση του τονικού ύψους περιορίζεται σε μονοφωνικά όργανα και εκτελέσεις τότε μπορεί να υπάρχει μία μόνο σωστή τιμή τονικού ύψους ανά χρονική στιγμή.

2. Από τη στιγμή που θα ξεκινήσει μία νότα δεν μπορούν τα στιγμιότυπα που την αποτελούν να έχουν διαφορετικό τονικό ύψος ή να μεσολαβούν κενά μεταξύ τους, πριν περάσουν 125ms περίπου (βάση του περιορισμού για το 1/16 που αναφέρθηκε παραπάνω, ενός μέσου BPM και του ρυθμού δειγματοληψίας του πίνακα στιγμιότυπων)

3. Δεν γίνεται εντός της διάρκειας μίας νότας να μεσολαβούν σφάλματα που επιστρέφει ο αλγόριθμος αυτοσυσχέτησης ή κενά.

Έτσι ακολουθούνται κάποιοι κανόνες κατά την σύγκριση των στιγμιότυπων που βασίζονται στις παραπάνω παραδοχές.



4.1 Σύγκριση των αποτελεσμάτων με και χωρίς τον αλγόριθμο αυτοδιόρθωσης ενεργοποιημένο

Στην παραπάνω εικόνα βλέπουμε με πράσινο τα στιγμιότυπα κατά την καταγραφή των οποίων ο αλγόριθμος αυτοδιόρθωσης ήταν ενεργοποιημένος και αχνά κόκκινα φαίνονται η επιπλέον πληροφορία που καταγράφηκε ενώ ο αλγόριθμος ήταν απενεργοποιημένος. Όπως βλέπουμε αποτελεσματικές διορθώσεις αυτού του αλγορίθμου δεν ξεπερνάνε το 20% επί των συνολικών λαθών, καθώς βρίσκεται ακόμα σε πολύ πρώιμο στάδιο και απλά υλοποιείται για να δόσει ένα στίγμα μελλοντικής δουλειάς.

4.4.6 Χαρακτηρισμός τον αντικειμένων με βάση τη μουσική πληροφορία των στιγμιότυπων

Ο αλγόριθμος αυτός είναι μία επέκταση του αλγόριθμου της διόρθωσης αλλά ταυτόχρονα είναι και ο τρόπος που δημιουργούνται τα αντικείμενα των νοτών. Ο χαρακτηρισμός βασίζεται στην αρχή και στο τέλος κάθε νότας με την λογική παραδοχή ότι κάθε αρχή νότας που εντοπίζεται έχει ένα τέλος και όλα τα ενδιάμεσα στιγμιότυπα έχουν το ίδιο τονικό ύψος. Αυτή η λειτουργία βασίζεται στην πληροφορία του τέλους κάθε νότας άρα δεν μπορεί να λειτουργήσει σε πραγματικό χρόνο καθώς αν μία νότα παίζεται σε μία συγκεκριμένη χρονική στιγμή δεν γνωρίζουμε εκ των προτέρων πότε θα τελειώσει. Έτσι αυτός ο αλγορίθμος εκτελείτε σειριακά στον πίνακα στιγμιότυπων με μία μικρή καθυστέρηση. Τα αποτελέσματα του καταγράφονται στο σύστημα αλλά μόνο κατ' απαίτηση του χρήστη αναπαριστώνται και αντικαθιστούν τα προηγούμενα. Πιο αναλυτικά, ο αλγόριθμος όταν εντοπίζει ότι ένα στιγμιότυπο είναι αρχή (onset) και ένα άλλο το τέλος (offset) μίας συγκεκριμένης νότας τότε βρίσκει το πιο κοινό τονικό ύψος μεταξύ των στιγμιότυπων και το ορίζει ως το τονικό ύψος της νότας. Τότε είναι η στιγμή που δημιουργείτε το αντικείμενο της νότας το οποίο απαρτίζεται από στιγμιότυπα. Μέσω του χαρακτηρισμού των αντικειμένων μπορούμε να διορθώσουμε τονικά ύψη αλλά και να διορθώσουμε στιγμιότυπα για τα οποία δεν έχουμε καθόλου πληροφορία μιας και ο αλγόριθμος αυτοσυσχέτησης τα αναγνώρισε ως κενά ή λάθη, φυσικά αν είναι εντός των ορίων μίας αρχής και ενός τέλους.

Ανίχνευση του τέλους (offset) μίας νότας

Μέσα από δοκιμές και παρατηρήσεις υλοποιήθηκε μία μέθοδος ανίχνευσης των offset η οποία θέτει ένα κατώφλι και με μία συγκεκριμένη σειρά συγκρίσεων του μήκους του κύματος του κάθε στιγμιότυπου ορίζει ένα στιγμιότυπο ως τέλος της νότας.

4.5 Σχεδιασμός υλοποίησης

4.5.1 Λογικός Σχεδιασμός

Η παρούσα έκδοση της εφαρμογής υλοποιεί ένα μέρος των αλγορίθμων που έχουν μελετηθεί. Αποφασίστηκε η παρουσίαση περιορισμένου εύρους λειτουργικοτήτων ώστε η εφαρμογή να είναι κατ 'αρχάς σταθερή, λειτουργική και να δίνει το στίγμα των μελλοντικών επιδιώξεων της.

Η χρήση της εφαρμογής από την μεριά των χρηστών περιγράφεται στην ακόλουθη λογική σειρά:

- Για να γίνει χρήση της εφαρμογής ο χρήστης θα πρέπει να έχει ένα τελευταίας τεχνολογίας φυλλομετρητή ο οποίος **να υποστηρίζει το Web Audio API.** Αφού πληκτρολογήσει το ανάλογο URL έχει πρόσβαση στην εφαρμογή από περιβάλλον desktop, tablet ή mobile.

- Σε πρώτη φάση **ο χρήστης επιλέγει το μουσικό όργανο** από το οποίο θέλει να ανακτήσει και να καταγράψει μουσική πληροφορία. Η επιλογή οργάνου γίνεται από μία προκαθορισμένη λίστα. Επίσης υπάρχει δυνατότητα να αλλάξει επιλογή αφού έχει ξεκινήσει η καταγραφή.

- Ακολούθως επιλέγει το μέσο από το οποίο θα γίνει η ανάκτηση. Υπάρχουν δύο επιλογές, το **μικρόφωνο** και το **ηχητικό απόσπασμα από αρχείο.** Η επιλογή του μικροφώνου συνεπάγεται με την άμεση εκκίνηση της ανάκτησης από την εφαρμογή. Αν επιλεγεί ένα μουσικό αρχείο πρώτα γίνεται φόρτωση του στο σύστημα και μετά ο χρήστης καλείτε να εκκινήσει την αναπαραγωγή του από το πλήκτρο **Play/Stop**, υπάρχει η δυνατότητα να σταματήματος της αναπαραγωγής από το ίδιο πλήκτρο.

- Λόγο των διαφορετικών συνθηκών που ισχύουν για κάθε υπολογιστικό σύστημα, κάθε μέσο ηχογράφησης αλλά και κάθε μουσικό όργανο, **ο χρήστης έχει τη δυνατότητα να αυξήσει την ένταση** του εισερχόμενου ηχητικού σήματος σε περίπτωση που η καταγραφή της μουσικής πληροφορίας φαίνεται ελλιπής.

- Ο **δείκτης τονικού ύψους** της εφαρμογής απεικονίζει κάθε στιγμή την τελευταία νότα που αναπαράχθηκε. Επίσης παρέχει μία ακόμα πληροφορία που έχει να κάνει με την ακρίβεια της νότας (cents), το κατά πόσο δηλαδή 'κουρδισμένη' ήταν η νότα αυτή.

- Ο **δείκτης των χτύπων ανά λεπτό** απεικονίζει το υπολογισμένο, βάσει των αλγορίθμων του συστήματος BPM. Η αρχική τιμή του κατά την εκκίνηση της εφαρμογής είναι ένα μέσο κοινό BPM της τάξεως των 120. Εδώ η πρώτη πραγματική τιμή που θα εξαχθεί και θα έχει υπολογιστεί από το μουσικό κομμάτι που αναπαράγεται θα έχει μία μικρή καθυστέρηση. Αυτό συμβαίνει διότι ο αλγόριθμος χρειάζεται έναν ελάχιστο αριθμό από onset για να κάνει τον υπολογισμό. Λόγο της επεξεργασίας των δεδομένων σε πραγματικό χρόνο ο αριθμός των χτύπων ανά λεπτό δύναται να εναλλάσσεται ανάλογα με την ταχύτητα του αναπαραχθέντος κομματιού κάθε στιγμή.

- Η **απεικόνιση της μουσικής πληροφορίας σε παρτιτούρα,** καταγράφει τις νότες που αναπαράγονται σε πεντάγραμμο και με τον κοινά αποδεκτό μουσικό συμβολισμό. Αναπαριστώνται σε συνάρτηση με την πληροφορία του χρόνου, το οποίο δίνει την δυνατότητα ένταξης των νοτών σε μέτρα , με συγκεκριμένη διάρκεια συμβολισμένες αναλόγως και απεικόνιση των παύσεων. Η απεικόνιση σε παρτιτούρα είναι ένα κομμάτι της εφαρμογής που βρίσκεται σε αρχικό στάδιο ανάπτυξης, υλοποιήθηκε κυρίως για να παρουσιάσει αυτή τη δυνατότητα.

- Τέλος, ο χρήστης έχει τη δυνατότητα να βλέπει σε πραγματικό χρόνο την **καταγραφή των αναπαραχθέντων νοτών σε ένα περιβάλλον αναπαράστασης MIDI**. Το κομμάτι αυτό αποτελεί και τη βασική λειτουργικότητα της εφαρμογής. Το Pianoroll είναι χωρισμένο σε νότες στον άξονα y και χρόνο/μέτρα στο άξονα x. Η απεικόνιση γίνεται με την μορφή ορθογωνίων σχημάτων τα οποία μεγαλώνουν ανάλογα με τη διάρκεια της εκάστοτε νότας σε πραγματικό χρόνο. Ο αρχικός σχεδιασμός του pianoroll κατά την εκκίνηση της εφαρμογής γίνεται με βάση το κοινά αποδεκτό BPM που αναφέρθηκε παραπάνω (*120 BPM*). Η εφαρμογή δείνει στο χρήστη κάποιες επιπλεόν επιλογές οι οποίες είναι η αυτόματη διόρθωση σε πραγματικό χρόνο (*Auto-correction*), ο επανασχεδιασμός με πλήρη διόρθωση (*Redraw corrected*), ο επανασχεδιασμός βάση του παρόντος BPM (*Redraw on BPM*), η εκκαθάριση του pianoroll (*Clear*), και η ενεργοποίηση της κίνησης του pianoroll ακλουθώντας την αναπαραγωγή (*Follow*). Η αυτόματη διόρθωση σε πραγματικό χρόνο (*Auto-correction*), επεξεργάζεται τη σχετικότητα μεταξύ των αναγνωρισθέντων γειτονικών τονικών υψών στο χρόνο και όταν είναι ενεργοποιημένη αποκλείει την απεικόνιση κάποιων νοτών ως αναγνωρισμένες για την συγκεκριμένη χρονική στιγμή βάση κάποιων κριτηρίων. Ο επανασχεδιασμός με πλήρη διόρθωση (*Redraw corrected*) είναι μία κατ απαίτηση λειτουργία η οποία επεξεργάζεται τα μέχρι τώρα δεδομένα και παρουσιάζει ένα διορθωμένο σύνολο των νοτών που καταγράφηκαν μέχρι την παρούσα χρονική στιγμή. Επεξεργάζεται περισσότερα δεδομένα από όσα είναι προσβάσιμα από τους αλγορίθμους σε πραγματικό χρόνο και εξάγει ένα πιο ασφαλές αποτέλεσμα. Οι νότες που διορθώνει ο αλγόριθμος αυτός αλλάζουν χρώμα ώστε να

αναγνωρίζονται. Ο επανασχεδιασμός βάση του παρόντος BPM (*Redraw on BPM*) είναι μία κατ απαίτηση λειτουργία η οποία επανασχεδιάζει το pianoroll λαμβάνοντας υπόψη τα μέχρι τώρα εξαχθέντα στοιχεία που είναι σχετικά με την ταχύτητα. Η λειτουργία αυτή υλοποιεί επίσης και τον επανασχεδιασμό με πλήρη διόρθωση.

4.5.2 Φυσικός Σχεδιασμός

Η δομή της εφαρμογής αποτελείτε από επιμέρους namespaces και κλάσεις οι οποίες έχουν διαχωριστεί ανάλογα με την λειτουργία που επιτελούν και παραθέτονται παρακάτω:

Namespaces:

- **Initializer**

  Ορίζει το αρχικό περιβάλλον της εφαρμογής καλώντας μεθόδους σχεδιασμού του περιβάλλοντος MIDI. Επίσης περιέχει κάποιες προεπιλογές σε μορφή ρυθμίσεων για διάφορα σημεία της εφαρμογής, οι οποίες μπορούν να κληθούν αργότερα από μεθόδους της εφαρμογής.

- **UI**

  Περιλαμβάνει όλες εκείνες τις μεθόδους οι οποίες κάνουν τη διάδραση του χρήστη με την εφαρμογή πραγματικότητα, από τον ορισμό των ενεργειών των κουμπιών του γραφικού περιβάλλοντος μέχρι τον επιλογέα αρχείων.

- **AudioUsher**

  Εδώ βρίσκεται ότι έχει να κάνει με το Web Audio API, η δημιουργία των AudioNodes και των ρυθμίσεων τους, ο ορισμός σύνθετων κόμβων καθώς και η υλοποίηση των συνδέσεων μεταξύ τους.

- **Actions**

  Το βασικότερο κομμάτι εδώ είναι το thread που εμπεριέχεται το οποίο στην ουσία καλύπτει το μεγαλύτερο ποσοστό των αναγκών της εφαρμογής. Από εδώ καλούνται οι μέθοδοι για την αναγνώριση των onset και του pitch, για τον υπολογισμό του tempo, για την απεικόνιση των αποτελεσμάτων κ.α. Επίσης περιλαμβάνει τις μεθόδους διόρθωσης, χαρακτηρισμού και κάποιες άλλες που έχουν να κάνουν με τη ακολουθία του συστήματος.

- **Timer**

  Εδώ εμπεριέχονται μέθοδοι οι οποίες κάνουν τον υπολογισμό και την απεικόνιση των αποτελεσμάτων σε πραγματικό χρόνο εφικτή. Πιο αναλυτικά υλοποιούνται συναρτήσεις που υπολογίζουν την σχετικότητα χρόνου και του χώρου του canva ώστε να υπάρχει παραλληλία καθώς και συναρτήσεις οι οποίες χρησιμεύουν για τον υπολογισμό χρόνων κύκλων και εκτέλεσης που εναλλάσσονται με στόχο τη διαφύλαξη ορθών αποτελεσμάτων.

- **Onset Detector**

  Προεπεξεργασία, μέθοδοι ανίχνευσης, μετά-επεξεργασία, επιλογή κορυφών, υπολογισμός tempo όλα αυτά είναι κομμάτια αυτού του namespace που συνθέτονται όλα μαζί σε μία κεντρική μέθοδο η οποία καθορίζει τη σειρά ακολουθίας και διαχειρίζεται τα αποτελέσματα.

- **Pitch Detector**

  Περιλαμβάνει της μεθόδους σχετικά με την εκτίμηση του τονικού ύψους, διαχειρίζεται τα αποτελέσματα καθώς και την επικοινωνία με τον Autocorrelate Worker που τρέχ3ει παράλληλα και υλοποιεί τον αλγόριθμο της αυτοσυσχέτησης στο εισερχόμενο μετασχηματισμένο ηχητικό σήμα.

- **Drawer**

  Οι μέθοδοι που απαρτίζουν αυτό το namespace έχουν να κάνουν με τη διάδραση της καταγραφής με τον canva, επίσης περιλαμβάνονται μέθοδοι που αρχικοποιούν το περιβάλλον απεικόνισης MIDI και παρτιτούρας.

Κλάσεις:

- **Note Snapshot**

  Σε κάθε κύκλο που πραγματοποιεί το βασικό thread καταγράφεται και ένα snapshot από τον Pitch Detector, αν αυτό το snapshot έχει αποδεκτή τονική πληροφορία τότε δημιουργείτε ένα αντικείμενο Note Snapshot από τον Pitch Detector με όλα τα χαρακτηριστικά που προκύπτουν από μεθόδους που τρέχουν παράλληλα.

- **Silence Snapshot**

Σε περίπτωση που για κάποιο snapshot ο Pitch Detector επιστρέψει τιμή κενού δημιουργείτε το ανάλογο αντικείμενο παύσης, με χρονικές και σχετικές με τον canva ιδιότητες.

- **Error Snapshot**

  Σε περίπτωση που για κάποιο snapshot ο Pitch Detector επιστρέψει μία συγκεκριμένη τιμή που εκλαμβάνεται ως σφάλμα δημιουργείτε το ανάλογο αντικείμενο σφάλματος, με χρονικές και σχετικές με τον canva ιδιότητες.

- **Single Note**

  Είναι είδος αντικειμένου που περιλαμβάνει ένα σύνολο από snapshots που συνθέτουν μια ολοκληρωμένη νότα. Δημιουργείτε μόλις εντοπιστεί ότι ένα snapshot είναι το τέλος μίας νότας. Επίσης περιλαμβάνει και ένα αντικείμενο νότας υλοποιημένο σύμφωνα με το API του Vexflow για υποστήριξη της απεικόνισης σε παρτιτούρα.

- **Notes Voice**

  Περιλαμβάνει όλες τα αντικείμενα Single Note που απαρτίζουν χρονικά ένα μέτρο το οποίο υπολογίζεται σύμφωνα με τους χτύπους ανά λεπτό. Είναι απαραίτητο για την απεικόνιση της μουσικής σημειογραφίας στην παρτιτούρα σύμφωνα με το Vexflow API.

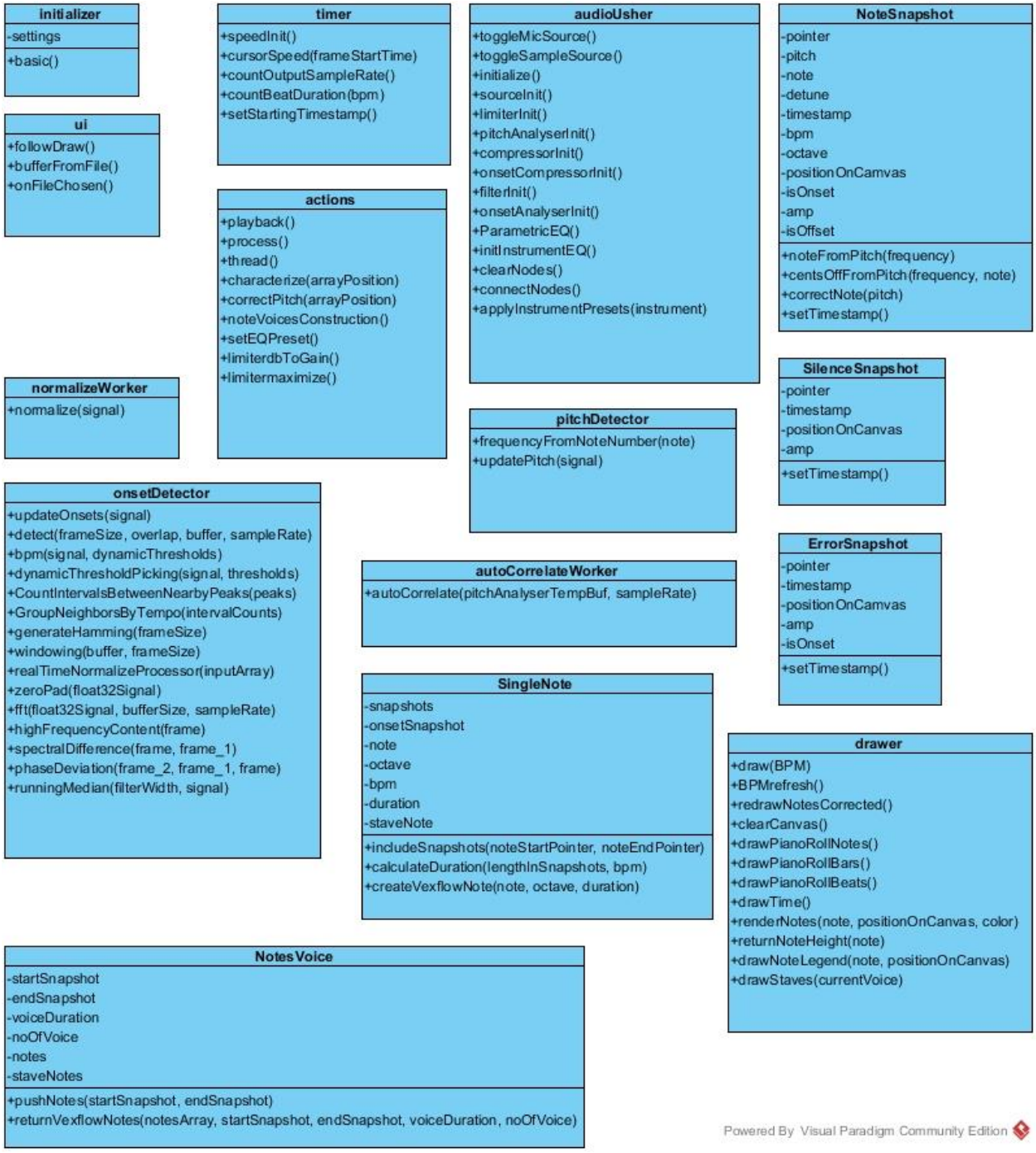
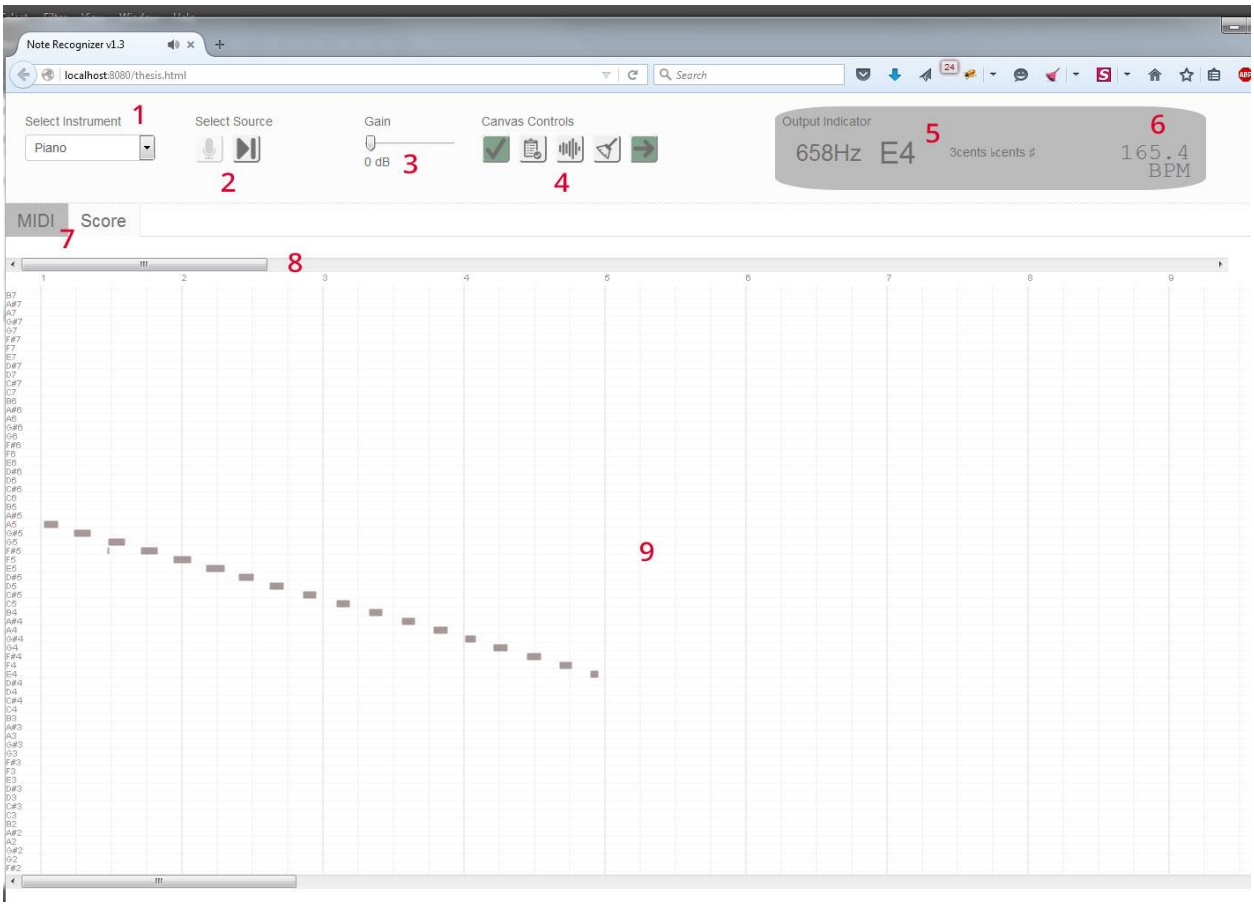


Figure 4.2 UML απεικόνιση των namespaces και κλάσεων με τα περιεχόμενα τους

4.6 Υλοποίηση

4.6.1 Εγχειρίδιο χρήσης

Η εφαρμογή Note Recognizer είναι πολύ απλή και εύκολη στη χρήση. Μπορεί να τρέξει σε όλους τους σύγχρονους browser και σε όλες τις συσκευές.



4.3 Η διεπαφή χρήστη της εφαρμογής, οι αριθμοί εξηγουνται παρακάτω

1. **Επιλογή οργάνου (Select Instrument)**

   Ο χρήστης επιλέγει το μουσικό όργανο από το οποίο θέλει να ανακτήσει και να καταγράψει μουσική πληροφορία.

2. **Επιλογή πηγής (Select Source)**

   Ο χρήστης επιλέγει το μέσο από το οποίο θα γίνει η ανάκτηση. Υπάρχουν δύο επιλογές, το μικρόφωνο και το ηχητικό απόσπασμα από αρχείο.

3. **Αύξηση έντασης (Gain)**

   Ο χρήστης έχει τη δυνατότητα να αυξήσει την ένταση (Gain) του εισερχόμενου ηχητικού σήματος σε περίπτωση που η καταγραφή της μουσικής πληροφορίας φαίνεται ελλιπής.

4. **Επιλογές σχεδίασης (Canvas Controls)**

   Η εφαρμογή δίνει στο χρήστη κάποιες επιπλέον επιλογές οι οποίες είναι η αυτόματη διόρθωση σε πραγματικό χρόνο (Auto-correction), ο επανασχεδιασμός με πλήρη διόρθωση (Redraw corrected), ο επανασχεδιασμός βάση του παρόντος BPM (Redraw on BPM), η εκκαθάριση του pianoroll (Clear), και η ενεργοποίηση της κίνησης του pianoroll ακλουθώντας την αναπαραγωγή (Follow).

5. **Δείκτης τονικού ύψους (Pitch Indicator)**

   Απεικονίζει πληροφορίες για την τελευταία νότα που αναπαράχθηκε.

6. **Δείκτης χτύπων ανά λεπτό (BPM Indicator)**

   Απεικονίζει το tempo του κομματιού που αναπαράγεται.

7. **Επιλογή αναπαράστασης (Notation Tabs)**

   Ο χρήστης μπορεί να επιλέξει αν θα βλέπει την καταγραφή σε περιβάλλον αναπαράστασης MIDI (*pianoroll*) ή σε παρτιτούρα.

8. **Μπάρα αναπαράστασης MIDI**

   Ο χρήστης μπορεί να ρυθμίσει τη θέση θέασης της καταγραφής.

9. **Καταγραφή μουσικής πληροφορίας (Notation)**

   Η απεικόνηση των καταγεγραμένων μουσικών πληροφοριών.

Βασικές οδηγίες

Για να καταγράψουμε ένα ηχητικό απόσπασμα πρέπει να έχουμε ένα μικρόφωνο ή ένα αρχείο ήχου. Επιλέγουμε το μουσικό όργανο αναπαραγωγής και την πηγή ήχου, πατάμε

αναπαραγωγή ή ξεκινάμε να παίζουμε και βλέπουμε τις νότες που αναπαράγονται να καταγράφονται.

Οδηγίες σχετικά με το μικρόφωνο

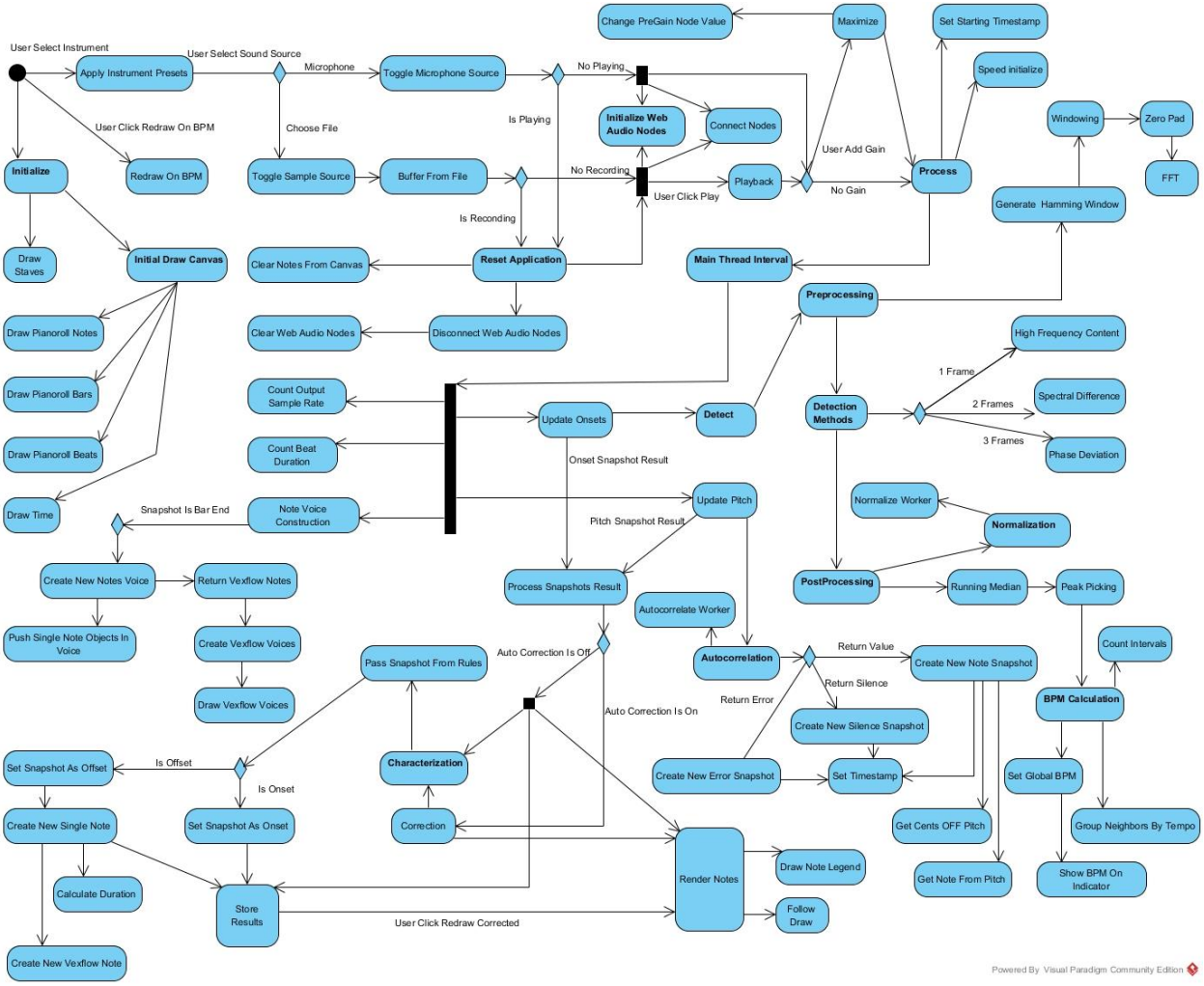
Εάν χρησιμοποιούμε μικρόφωνο φροντίζουμε να το έχουμε κοντά στην πηγή του ήχου. Αν η ένταση είναι χαμηλά ή τα αποτελέσματα που καταγράφονται φαίνονται ελλείπή τότε αυξάνουμε την ένταση (*gain*).

Επίπεδο ασφάλειας αποτελέσματος

Η εφαρμογή βρίσκεται σε πειραματικό στάδιο οπότε προτείνεται τα αποτελέσματα να μην χρησιμοποιούνται για σοβαρούς σκοπούς.

4.6.2 Εγχειρίδιο συστήματος

Οι παρακάτω ενότητες αναλύουν την λειτουργία της εφαρμογής σε τεχνικό επίπεδο.



4.4 Διάγραμμα δραστηριοτήτων της εφαρμογής (Activity diagram)

4.6.2.1 Βιβλιοθήκες που χρησιμοποιήθηκαν (*Libraries*)

Η εφαρμογή χρησιμοποιεί τις παρακάτω βιβλιοθήκες:

- **Jquery**

  Χρησιμοποιείται για διεργασίες που έχουν να κάνουν με τη διεπαφή χρήστη.

- **Bootstrap**

  Χρησιμοποιούνται κάποια πρότυπα στη διεπαφή χρήστη.

- **Complex.js**

  Είναι απαραίτητη γιατί ο μετασχηματισμός FFT που υλοποιείτε κατά τον εντοπισμό των onset στο στάδιο της προεπεξεργασίας του σήματος εξάγει πολύπλοκους αριθμους.

- **Dsp.js**

  Απαραίτητη βιβλιοθήκη για την υλοποίηση πολλών αλγορύθμων που έχουν να κάνουν με τον εντοπισμό των onset, στο στάδιο προεπεξεργασίας και συναρτήσεων εντοπισμού.

- **Vexflow**

  Βιβλιοθήκη χρησιμοποιηται για την απεικόνηση της μουσικής σημειογραφίας σε παρτιτούρα.

Κάποιες από τις παραπάνω βιβλιοθήκες με την χρήση του εργαλείου browserify μετατράπηκαν ώστε να εκτελούνται στον φυλλομετρητή του πελάτη, αφού είναι φτιαγμένες για να λειτουργού σε περιβάλλον node.js.

4.6.2.2 Προετοιμασία διεπαφής, μεταβλητών και προεπιλογών (*Initialize*)

Στο αρχείο contructor.js αρχικοποιούνται όλες οι global μεταβλητές, πίνακες, αντικείμενα, namespaces και workers που χρησιμοποιούνται στην εφαρμογή. Το αρχείο initializep.js περιέχει όλα τις προεπιλογές που θα χρησιμοποιηθούν αργότερα από τον compressorNode και το parametricEQ. Επίσης μόλις η εφαρμογή φορτώσει καλούνται οι απαραίτητες συναρτήσεις ώστε να σχεδιαστεί το περιβάλλον απεικόνισης MIDI και

παρτιτούρας. Οι συναρτήσεις που χρησιμοποιούνται για την αρχική αναπαράσταση βρίσκονται στο αρχείο drawer.js και είναι οι drawPianoRollNotes, drawPianoRollBars, drawPianoRollBeats, drawTime. Η συνάρτηση drawPianoRollBeats και drawPianoRollBars χρησιμοποιούν το μέγεθος εύρος χτύπου (beat) που έχουν αρχικά οριστεί. Η συνάρτηση drawTime υπολογίζει το μήκος ενός δευτερολέπτου βάση των BPM και του μήκους ενός beat.

```
drawTime: function () {
    pianoRollContext.fillStyle = "#a29292";
    pianoRollContext.font = "10px Arial";
    pianoRollContext.lineWidth = 1;
    pianoRollContext.strokeStyle = '#D5D5D5';
    var initialMargin = 40;
    // the width of 1 second in pianoroll
    secWidth = BPM / 60 * beatWidth || 120 / 60 * beatWidth;
    var seconds = 0.00;
    var minutes = 0.00;
    for (var i = 0; i < pianoRollCanvas.width; i = i + secWidth) {
        if (seconds + minutes > minutes + 0.60) {
            minutes += 1.00;
            seconds = 0.00;
        }
        if (i === 0) {
            pianoRollContext.fillText((parseFloat(minutes) + parseFloat(seconds)).toFixed(2), i + initialMargin, 1000);
            pianoRollContext.beginPath();
            pianoRollContext.moveTo(i + initialMargin, 982);
            pianoRollContext.lineTo(i + initialMargin, 990);
        } else {
            pianoRollContext.fillText((parseFloat(minutes) + parseFloat(seconds)).toFixed(2), i + initialMargin - i / barWidth, 1000);
            pianoRollContext.beginPath();
            pianoRollContext.moveTo(i + initialMargin - i / barWidth, 982);
            pianoRollContext.lineTo(i + initialMargin - i / barWidth, 990);
        }
        pianoRollContext.stroke();
        seconds += 0.01;
    }
},
```

4.5 Η υλοποίηση της συνάρτησης drawTime

4.6.2.3 Διεπαφή χρήστη (*User Interface*)

Το αρχείο ui.js υλοποίει όλους τους listeners για τα κουμπιά της διεπαφής χρήστη, κατ' εξοχήν ο κώδικάς που υλοποιείται είναι βασισμένων στο jquery. Επίσης από εδώ διαχειρίζεται και η λειτουργία της επιλογής αρχείου από το χρήστη. Η συνάρτηση onFileChosen που καλείται μόλις ο χρήστης επιλέξει ένα αρχείο καλεί τη συνάρτηση bufferFromFile η οποία και ξεκινάει τις βασικές διεργασίες του Web Audio API.

```
// Creates a buffer from a file, callback is a function of the newly created buffer
bufferFromFile: function (file, callback) {
    var reader = new FileReader();
    reader.onload = function (e) {
        audioContext.decodeAudioData(e.target.result, function (buffer) {
            console.log("Sound loaded: " + file.name);
            theBuffer = buffer;
            audioUsher.webAudioInit.processNodes.initialize();
            audioUsher.connector.connectNodes();
            callback(buffer);
        });
    };
    reader.readAsArrayBuffer(file);
},
onFileChosen: function (e) {
    // Disable the playback buttons during loading
    $("#play-button").hide();
    document.getElementById("play-button").disabled = true;
    // Set the source buffer to be that of the chosen file
    var file = e.target.files[0];
    ui.bufferFromFile(file, function (buffer) {
        $("#play-button").show();
        $("#sample-button").hide();
        document.getElementById("play-button").disabled = false;
    });
}
```

4.6 Υλοποίηση των συναρτήσεων onFileChosen και bufferFromFile

4.6.2.4 Web Audio API

Όλες οι διεργασίες που έχουν να κάνουν με το Web Audio API υλοποιούνται στο namespace AudioUsher το οποίο περιλαμβάνεται στο αρχείο web_audio_nodes.js.

- **Toggle Source**

  Είναι οι βασικές συναρτήσεις που αναλαμβάνουν τη διαχείριση των AudioNodes.

- **Initialize**

  Δημιουργεί όλα τα AudioNodes

- **clearNodes**

  Μηδενίζει τις μεταβλητές των AudioNodes για περιπτώσεις εναλλαγής πηγής.

- **connectNodes**

  Συνδέει όλα τα AudioNodes στην επιθυμητή σειρά μεταξύ τους.

```
connector: {
    connectNodes: function () {
        if (micSelected) {
            mediaStreamSource.connect(compressorNode);
        }
        else if (sampleSelected) {
            sourceNode.connect(compressorNode);
        }
        // compressor
        compressorNode.connect(instrumentEQNodes.input);
        // EQ
        instrumentEQNodes.output.connect(preGainNode);
        //limiter
        preGainNode.connect(limiterNode);
        //1out
        limiterNode.connect(pitchAnalyserNode);
        //2out
        limiterNode.connect(onsetCompressorNode);
        //onsetCompressor
        onsetCompressorNode.connect(onsetAnalyserNode);
        //3out
        limiterNode.connect(audioContext.destination);
    }
},
```

4.7 Υλοποίηση της συνάρτησης connectNodes - Η σειρά που είναι συνδεδεμένα τα AudioNodes

4.6.2.5 Νήμα εκτέλεσης (*Thread*)

Για να υλοποιηθεί το νήμα εκτέλεσης χρησιμοποιήθηκε η συνάρτηση setInterval() με καθυστέρηση επανεκτέλεσης 20ms. Για την εφαρμογή θα ήταν πολύ χρήσιμο αν το νήμα εκτέλεσης είχε μεγαλύτερη συχνότητα. Υπάρχει αυτή η δυνατότητα, όμως όταν ορίζεται μεγαλύτερη συχνότητα δεν είναι πραγματική γιατί μπορεί να ξεκινήσει όπως ορίζεται αλλά από κύκλο σε κύκλο μειώνεται και έτσι χάνει τη σταθερότητα της η εφαρμογή.

4.6.2.6 Συσχετιστές χρόνου και Μετρητές

Οι συσχετιστές παίζουν πολύ πρωταγωνιστικό ρόλο σε αυτή την εφαρμογή γιατί είναι αυτοί που στην ουσία σχετίζουν το χρόνο με το χώρο εντός της εφαρμογής. Το αρχείο timer.js περιέχει αυτές της συναρτήσεις μέτρησης. Βασική συνάρτηση για την απεικόνιση σε πραγματικό χρόνο, ώστε το σύστημα να καταγράφει με μία ανάλογη των BPM ταχύτητα, είναι η speedInit. Αυτή η συνάρτηση στην ουσία υπολογίζει ανά πόσα ms που περνάνε ο καταγραφέας πρέπει να αλλάζει pixel. Προς το παρόν η συνάρτηση αυτή χρησιμοποιείται μόνο για την αρχικοποίηση της ταχύτητας όχι για την αλλαγή της.

```
speedInit: function () {
    BPS = BPM / 60;
    //pixel speed per second
    PSPS = BPS * beatWidth;
    // for example in 120 bpm it moves 1 pixel every 25ms
    //milliseconds pass per pixel
    MSPPP = 1 / (PSPS / 1000);
    // dynamic millisecond difference
    dynamicMsDiff = MSPPP / 1000;
    updatePitchStartTime = audioContext.currentTime;
},
```

4.8 Υλοποίηση της συνάρτησης speedInit

Στο ίδιο αρχείο υλοποιείται η συνάρτηση cursorSpeed, η οποία ορίζει τον ρυθμό καταγραφής, δηλαδή σε κάθε κύκλο ξέρει σε ποιο pixel στον άξονα x η καταγραφή βρίσκεται. Από την πλευρά των μετρητών υπάρχουν οι συναρτήσεις countOutputSampleRate και countBeatDuration, που αντίστοιχα υπολογίζουν το ρυθμό δειγματοληψίας του πίνακα στιγμιότυπων και την αντιστοιχία διάρκειας νότας και στιγμιότυπων.

```
// this is to disallow change pixel faster than MSPPP
cursorSpeed: function (frameStartTime) {
    if (frameStartTime - updatePitchStartTime >= dynamicMsDiff) {
        dynamicMsDiff = dynamicMsDiff + MSPPP / 1000;
        positionInCanvas++;
        return true;
    } else {
        return false;
    }
},
// this maybe varies because the setInterval has not a standard fps
countOutputSampleRate: function () {
    if (Date.now() - startProccessingTimestamp >= 1000) {
        finalOutputSampleRate = sumOutput.signal.length - tmpFinalOutputSamplesLength;
        startProccessingTimestamp = Date.now();
        tmpFinalOutputSamplesLength = sumOutput.signal.length;
        // here we can count the most common finalOutputSampleRate each time (in compare with the previous ones) and use this
    }
},
countBeatDuration: function (bpm) {
    // multiply fps with seconds of a minute to find sanpshots per minute
    // (every SingleNote has a certain amount of snapshots so we can determine its beat division)
    var minuteSnapshots = finalOutputSampleRate * 60;
    // how many snapshots is a beat
    var snapshotsPerBeat = minuteSnapshots / bpm;
    snapshotsBarDuration = snapshotsPerBeat * 4;
    return snapshotsPerBeat;
},
```

4.9 Υλοποίηση μερικών συναρτήσεων του namespace timer

4.6.2.7 Εντοπισμός onsets

Η συνάρτηση εντοπισμού των onset καλείται από το νήμα εφαρμογής σε κάθε κύκλο. Περιέχει μία βασική μέθοδο η οποία δένει όλα μαζί τα επιμέρους κομμάτια / βήματα του αλγορίθμου που εμπεριέχονται στο namespace onsetDetector.

```
detect: function (frameSize, overlap, buffer, sampleRate) { //7
    var numberOfFrames = Math.floor(buffer.length / frameSize);
    hammingWindow = this.preprocessing.generateHamming(frameSize);
    frameStart = 0;
    var windowedFrame = [];
    var fftFrame = null;
    var detectFrame = null;
    var tempFrame = null;
    var tempFrame_2 = null;
    for (var frameNumber = 0; frameNumber < numberOfFrames; frameNumber++) {
        // PRE-PROCESSING
        windowedFrame = this.preprocessing.windowing(buffer, frameSize);
        var zeroPaddedFrame = this.preprocessing.zeroPad(windowedFrame);
        tempFrame_2 = tempFrame;
        tempFrame = fftFrame;
        // in order to achive zero padding the fft frame must be bigger than the input frame
        // so i multiply the fft frame with 2 and i add zeros to the second half of the signal
        fftFrame = this.preprocessing.fft(zeroPaddedFrame, frameSize * 2, sampleRate);
        // DETECTION FUNCTIONS
        if (tempFrame === null) {
            detectFrame = null;
            detectFrame = this.detectionMethods.highFrequencyContent(fftFrame);
        } else if (tempFrame_2 === null && tempFrame !== null) {
            detectFrame = this.detectionMethods.spectralDifference(fftFrame, tempFrame);
        } else if (tempFrame_2 !== null && tempFrame !== null) {
            detectFrame = this.detectionMethods.phaseDeviation(tempFrame_2, tempFrame, fftFrame);
        }
        frameStart = frameStart + frameSize - (frameSize * overlap);
    }
    // import the values in an unnormalized array to avoid renormalize of the old array
    unNormalizedOutput.push(detectFrame);
    // POST-PROCESSING
    // NORMALIZATION
    normalizeWorker.postMessage({array: JSON.stringify(unNormalizedOutput), type: "last"});
    finalOutput = normalizedOutput.lastNormalize;
    // RUNNING/MOVING MEDIAN - THRESHOLDING - https://github.com/mikolalysenko/moving-median
    dynamicThresholds = onsetDetector.postprocessing.runningMedian(3, finalOutput);
    BPM = this.bpm(finalOutput, dynamicThresholds);
    document.getElementById('bpm-indicator').innerHTML = "<kbd>" + BPM + " BPM</kbd>";
},
```

4.10 Υλοποίηση της συνάρτησης detect που κάνει τη διαχείριση του αλγορίθμου

Η μέθοδος εντοπισμού που επιλέγεται βασίζεται στο μέγεθος του αρχικού παραθύρου που θα χρησιμοποιήσουμε, επειδή ορισμένες μέθοδοι χρειάζονται παραπάνω από ένα παράθυρα για να κάνουν τον εντοπισμό. Η προεπιλογή της εφαρμογής είναι ο αλγόριθμος HFC, βάσει του ορισμένου μεγέθους παραθύρου.
Κατά τη διάρκεια της μετά-επεξεργασίας σε κάθε κύκλο πρέπει να γίνεται κανονικοποίηση τιμών, η οποία αποφασίστηκε αν υλοποιείται μέσα σε ένα worker ο οποίος τρέχει παράλληλα. Σαν σημείο του αλγορίθμου με εξέχουσα σημαντικότητα ορίζεται η υλοποίηση του φίλτρου μέσης τιμής, το οποίο καθορίζει τα κατώφλια και υλοποιείται στη συνάρτηση runningMedian. Κατά την επιλογή κορυφών η συνάρτηση dynamicThresholdPicking η οποία υλοποιεί τις θεωρητικές προσεγγίσεις που αναλύθηκαν στο κεφάλαιο 3.

```
dynamicThresholdPicking: function (signal, thresholds) {
    var onsetArray = [];
    var thresholdedSignal = [];
    for (var i = 0; i < signal.length; ++i) {
        if ((signal[i] > thresholds[i]) && (signal[i - 1] <= signal[i] <= signal[i + 1])) {
            thresholdedSignal[i] = 1000;
            onsetArray.push(i);
            // to avoid map as onset the second occuring event in the same note which pass these cor
            i = i + 10;
        }
        else
            thresholdedSignal[i] = 0;
    }
    return {onsetArray: onsetArray, thresholdedSignal: thresholdedSignal};
},
```

4.11 Υλοποίηση της συνάρτησης dynamicThresholdPicking

4.6.2.8 Εκτίμηση τονικού ύψους

Η συνάρτηση updatePitch καλείται από το νήμα εφαρμογής σε κάθε κύκλο. Αρχικά στέλνει κάθε κομμάτι του σήματος στον worker που υλοποιεί τον αλγόριθμο της αυτοσυσχέτησης. Στη συνέχεια ανάλογα με την τιμή που θα επιστρέψει ο worker η συνάρτηση δημιουργεί αντικείμενα στιγμιότυπων παύσης, λάθους ή νότας.

```
updatePitch: function (signal) {  //5
    if (timer.cursorSpeed(audioContext.currentTime)) {
        try {
            autoCorrelateWorker.postMessage({array: signal, samples: audioContext.sampleRate});
        } catch (err) {
        }
        // This is the silence which autocorrelation understands
        // noteEnded is asynchronous so it lose the two snapshots after the offset
        // ( i use for this -> actions.correction.subroutines.silenceNextTwoSnapshots)
        if (ac === -1 || noteEnded) {
            // create silcence object instance
            var silence = new SilenceSnapshot(sonicScope.length);
            sonicScope.push(silence);
            detectorElem.className = "vague";
            $("#pitch").text("--");
            pitchElem.innerText = "--";
            detuneElem.className = "";
            detuneAmount.innerText = "--";
        } else if (ac === 11025) {
            // create silcence object instance
            var error = new ErrorSnapshot(sonicScope.length);
            sonicScope.push(error);
            detectorElem.className = "vague";
            $("#pitch").text("--");
            detuneElem.className = "";
            detuneAmount.innerText = "--";
        }
        // this condition applied only when autocorrelation understands a note
        else {
            detectorElem.className = "confident";
            pitch = ac;
            $("#pitch").text(Math.floor(pitch));
            // create note object instance
            var note = new NoteSnapshot(pitch, sonicScope.length);
            // this is a fulltime array which stores anything happens (silence or note)
            sonicScope.push(note);
            noteElem.innerHTML = note.note + note.octave;
```

4.12 Μέρος της υλοποίησης της συνάρτησης updatePitch

4.6.2.9 Αντικείμενα στιγμιότυπων

Η συνάρτηση updatePitch καλεί τους constructors των αντικειμένων NoteSnapshot, SilenceSnapshot και ErrorSnapshot. Το κάθε είδος αντικειμένου έχει κάποια αναγνωριστικά τα οποία παίρνουν τιμές από τη δημιουργία του αντικειμένου ή μέσα από μία άλλη συνάρτηση της εφαρμογής στη συνέχεια.

```
//note snapshots
function NoteSnapshot(pitch, notePointer) {
    this.pointer = notePointer;
    this.pitch = pitch;
    try {
        this.note = noteStrings[this.noteFromPitch(this.pitch) % 96].match(/[A-G+#]/gi).join('');
    } catch (err) {
    }
    this.detune = this.centsOffFromPitch(this.pitch, this.noteFromPitch(this.pitch));
    this.timestamp = this.setTimestamp();
    this.bpm = BPM;
    try {
        this.octave = noteStrings[this.noteFromPitch(this.pitch) % 96].match(/[0-9]+/g).map(function (n)
        {//just coerce to numbers
            return +(n);
        })[0];
    } catch (err) {
    }
    this.positionOnCanvas = positionInCanvas;
    this.isOnset = false;
    // the amplitude after the onset detection function
    this.amp = sumOutput.signal[this.pointer];
    this.isOffset = false;
```

4.13 Ο constructor της κλάσης NoteSnapshot με όλα της τα γνωρίσματα.

4.6.2.10 Πίνακας στιγμιότυπων

Η μουσική πληροφορία του ηχητικού σήματος εκφράζεται από τον πίνακα στιγμιότυπων του. Έτσι σε κάθε κύκλο όλα τα αποτελέσματα καταχωρούνται σε ένα κεντρικό πίνακα τον sumOutput. Τα περιεχόμενα αυτού του πίνακα καθορίζονται από το βασικό νήμα εκτέλεσης.

```
//onsetdetector
try {
    onsetAnalyserNode.getFloatTimeDomainData(onsetAnalyserBuffer);
    onsetDetector.updateOnsets(onsetAnalyserBuffer);
    var onsetDetectorResult = finalOutput[finalOutput.length - 1];
    // add onsetDetector results to a central array
    sumOutput.signal.push(onsetDetectorResult);
} catch (err) {
}
//pitchdetector
try {
    pitchAnalyserNode.getByteTimeDomainData(pitchAnalyserTempBuf);
    pitchDetector.updatePitch(pitchAnalyserTempBuf);
    var pitchDetectorResult = sonicScope[sonicScope.length - 1];
    // add pitchDetector results to a central array
    sumOutput.is.push(pitchDetectorResult);
} catch (err) {
}
```

4.14 Απόσπασμα κώδικα στο οποίο υλοποιείται η ώθηση των τιμών επιστροφής στον πίνακα στιγμιότυπων

4.6.2.11 Αυτοδιόρθωση (*Auto-Correction*)

Η συνάρτηση correctPitch εξετάζει τη μουσική πληροφορία του στιγμιότυπου της προπροηγούμενης νότας και των δύο επόμενων και προηγούμενων αυτής με σκοπό να διορθώσει τυχόν ασυνέχειες που θα εντοπίσει. Η συνάρτηση υλοποίει κάποιους κανόνες οι οποίοι βασίζονται στις παραδοχές που αναφέρθηκαν σε προηγούμενο κεφάλαιο. Τα υπό εξέταση στιγμιότυπα είναι όλα περιεχόμενα του πίνακα sumOutput.

```
var errCons = {
    lowthres: thisObject.pitch < lowThreshold,
    highthres: thisObject.pitch > highThreshold,
    prevpitchdiff: prevObject.pitch > thisObject.pitch + lowDiff,
    nextpitchdiff: nextObject.pitch > thisObject.pitch + lowDiff,
    prevcomp: thisObject.pitch !== prevObject.pitch,
    prev2comp: thisObject.pitch !== prev2Object.pitch,
    nextcomp: thisObject.pitch !== nextObject.pitch,
    next2comp: thisObject.pitch !== next2Object.pitch,
    prevandnextequal: prevObject.pitch === prev2Object.pitch === nextObject.pitch === next2Object.pitch,
    issilencebefore: previousAreSilence(arrayPosition - 3),
    hasspaceinnote: (prevObject instanceof SilenceSnapshot) && !(previousAreSilence(arrayPosition - 3)),
    // this can be true only for exact numbers not even a cent up/down
    isharmonic: (thisObject.pitch % nextObject.pitch) === 0,
    iserror: (thisObject instanceof ErrorSnapshot),
    prevnote: (prevObject instanceof NoteSnapshot),
};
result[1] = check(errCons.issilencebefore && errCons.nextcomp && errCons.next2comp);
result[2] = check(errCons.nextpitchdiff);
result[3] = check(errCons.hasspaceinnote);
result[4] = check(errCons.prevandnextequal && errCons.prevcomp);
result[5] = check(errCons.prevcomp && errCons.nextcomp && errCons.prevpitchdiff);
result[6] = check(errCons.lowthres && errCons.nextcomp && errCons.prevcomp);
result[7] = check(errCons.isharmonic && errCons.nextcomp);
result[8] = check(errCons.iserror && errCons.prevnote && !(errCons.issilencebefore));
```

4.15 Κριτήρια τα οποία καθορίζουν τη διόρθωση της μουσικής πληροφορίας των στιγμιότυπων

4.6.2.12 Απεικόνιση στιγμιότυπων (*Render notes*)

Την απεικόνιση των στιγμιότυπων υλοποιεί η συνάρτηση renderNotes η οποία βρίσκεται στο namespace drawer και καλείτε από το νήμα εκτέλεσης. Συγκρίνοντας το όνομα και την οκτάβα της νότας εντοπίζει σε ποιο ύψος του canva η νότα πρέπει να τυπωθεί. Το μήκος του canva στο οποίο πρέπει να τυπωθεί καθορίζεται από το τρέχον αντικείμενο NoteSnapshot.

4.6.2.13 Χαρακτηρισμός (*Characterization*)

Η συνάρτηση characterize τρέχει με μία καθυστέρηση 25 στιγμιότυπων από το τρέχον στιγμιότυπο, δηλαδή σε κάθε κύκλο το στιγμιότυπο που επεξεργάζεται είναι το characterizedSnapshot το οποίο είναι 25 θέσεις πιο πριν από το τρέχον στον πίνακα στιγμιότυπων sumOutput. Η συνάρτηση συγκρίνει τις τιμές των στιγμιότυπων με αυτές των onset από τον πίνακα sumOutput, έτσι ορίζει ένα στιγμιότυπο σαν onset. Επίσης μέσω μίας σύγκρισης πληροφορίας πλάτους κύματος των στιγμιότυπων που υλοποιείται στην σχέση isOffsetCondition ορίζονται τα offset. Μετά τους χαρακτηρισμούς onset και offset η ίδια συνάρτηση κάθε φορά που εντοπίζει offset ακολουθεί κάποια βήματα, αρχικά για να βρεί το προηγούμενο onset και στη συνέχεια ώστε να εντοπίσει τα πιο συχνά pitch και BPM των στιγμιότυπων μεταξύ των ορίων. Στη συνέχεια εξομοιώνει όλα τα συμπεριλαμβανόμενα στιγμιότυπα με τις τιμές αυτές και δημιουργεί ένα αντικείμενο SingleNote που τα συμπεριλαμβάνει όλα αυτά.

```
if (thisObject.isOffset) {
    var position = thisObject.pointer;
    var snapshotPitches = [];
    var snapshotBPM = [];
    position = thisObject.pointer;
    do {
        if (sumOutput.is[position] instanceof NoteSnapshot) {
            snapshotPitches.push(sumOutput.is[position].pitch);
            snapshotBPM.push(sumOutput.is[position].BPM);
        }
        position--;
    }
    while (!sumOutput.is[position].isOnset)
    var mostCommonPitch = initializer.tools.getMaxOccurrence(snapshotPitches);
    var mostCommonBPM = initializer.tools.getMaxOccurrence(snapshotBPM);
    position = thisObject.pointer;
    do {
        var tempPos = sumOutput.is[position].positionOnCanvas;
        var tempPointer = sumOutput.is[position].pointer;
        var tempTimestamp = sumOutput.is[position].timestamp;
        var tempIsOnset = sumOutput.is[position].isOnset;
        var tempIsOffset = sumOutput.is[position].isOffset;
        sumOutput.is[position] = new NoteSnapshot(mostCommonPitch, tempPointer);
        sumOutput.is[position].positionOnCanvas = tempPos;
        sumOutput.is[position].timestamp = tempTimestamp;
        sumOutput.is[position].isOnset = tempIsOnset;
        sumOutput.is[position].isOffset = tempIsOffset;
        sumOutput.is[position].BPM = mostCommonBPM;
        position--;
    }
    while (!sumOutput.is[position].isOnset)
    // to change the onset too which is the last
    var tempPos = sumOutput.is[position].positionOnCanvas;
    var tempPointer = sumOutput.is[position].pointer;
    var tempTimestamp = sumOutput.is[position].timestamp;
    var tempIsOnset = sumOutput.is[position].isOnset;
    var tempIsOffset = sumOutput.is[position].isOffset;
    sumOutput.is[position] = new NoteSnapshot(mostCommonPitch, tempPointer);
    sumOutput.is[position].positionOnCanvas = tempPos;
    sumOutput.is[position].timestamp = tempTimestamp;
    sumOutput.is[position].isOnset = tempIsOnset;
    sumOutput.is[position].isOffset = tempIsOffset;
    sumOutput.is[position].BPM = mostCommonBPM;
    //create the whole single note from the snapshots
    var note = new SingleNote(tempOnset.pointer, tempOffset.pointer);
    notes.push(note);
}
```

4.16 Κομμάτι της συνάρτησης characterize στο οποίο τίθενται τα όρια, γίνονται διορθώσεις και δημιουργείτε το αντικείμενο

4.6.2.14 Αντικείμενα Νοτών (*Single Notes*)

Ο constructor του αντικειμένου SingleNote παίρνει σαν ορίσματα τη θέση στον πίνακα sumOutput του στιγμιότυπου onset και την αντίστοιχη θέση του offset. Συμπεριλαμβάνει όλα τα στιγμιότυπα που το απαρτίζουν σε ένα πίνακα και αντλεί κάποια βασικά του γνωρίσματα από το πρώτο στιγμιότυπο που το απαρτίζει (όλα πλέον έχουν την ίδια πληροφορία). Επίσης μέσα από κάποιες συναρτήσεις σχετικότητας χρόνου που αναλύθηκαν πιο πριν υπολογίζετε η διάρκεια της νότας βάση του πλήθους των στιγμιότυπων.

```
SingleNote.prototype.calculateDuration = function (lengthInSnapshots, bpm) {
    snapshotsPerBeat = timer.countBeatDuration(bpm);
    var divisionCount = snapshotsPerBeat / lengthInSnapshots;
    if (divisionCount <= 1.20)
        return noteDurations[0];
    // whole note
    else if (divisionCount > 1.20 && divisionCount <= 3)
        return noteDurations[1];
    // half note
    else if (divisionCount > 3 && divisionCount <= 5)
        return noteDurations[2];
    else if (divisionCount > 5 && divisionCount <= 11)
        return noteDurations[3];
    else if (divisionCount > 11 && divisionCount <= 24)
        return noteDurations[4];
    else if (divisionCount > 24)
        return noteDurations[5];
    var tmpDifferences = [];
    for (var k = 0; k < noteDurations.length; k++) {
        // find the differences - the min diff is the index we need
        tmpDifferences[k] = Math.abs(durationOfDividedRests - noteDurations[k].math);
    }
};
SingleNote.prototype.createVexflowNote = function (note, octave, duration) {
    var vexflowNote = new Vex.Flow.StaveNote({keys: [note.toLowerCase() + '/' + octave.toString()], duration: duration.string});
    if (note.substr(1, 1) === '#') {
        // create sharp notes - it needs to be in different way
        vexflowNote.addAccidental(0, new Vex.Flow.Accidental("#"));
    }
    return vexflowNote;
};
```

4.17 Μέρος του κώδικά που περιγράφει την κλάση SingleNote

Τέλος, όπως βλέπουμε και παραπάνω το αντικείμενο περιέχει συνάρτηση ώστε να δημιουργήσει μία νότα Vexflow ώστε να χρησιμοποιηθεί αργότερα στην απεικόνιση παρτιτούρας.

4.6.2.15 Αντικείμενα μέτρων (*Note Voices*)

Λόγο ιδιαιτερότητας του Vexflow API καλούμαστε να τυπώνουμε τις νότες στην παρτιτούρα σε υποσύνολα τα οποία λέγονται voices. Έτσι η εφαρμογή υλοποιεί πρώτα ενα αντικείμενο το οποίο αντιπροσωπεύει ένα μέτρο και τοποθετεί εκεί αντικείμενα του τύπου SingleNote που έχουν ήδη δημιουργηθεί. Εντός του αντικειμένου υπάρχει η returnVexflowNotes η οποία δημιουργεί το Vexflow voice, τοποθετεί τις νότες εντός του και το τυπώνει.

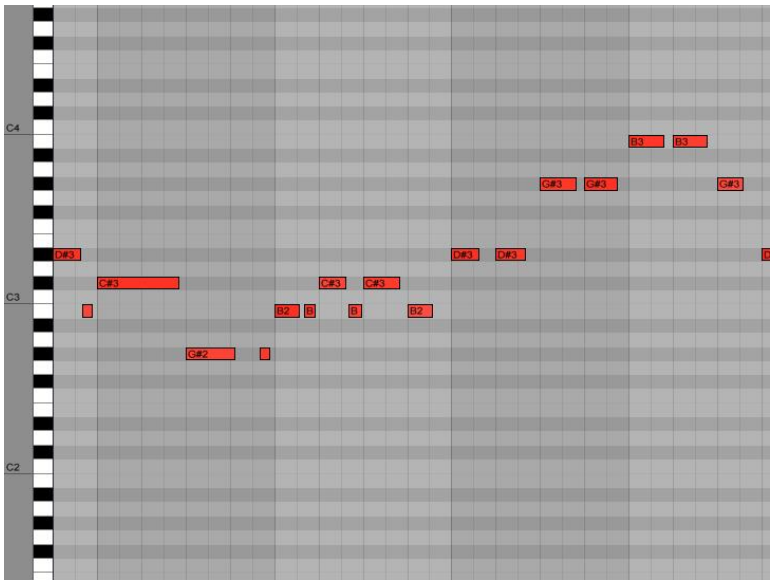
5 Αποτελέσματα

5.1 Δοκιμές

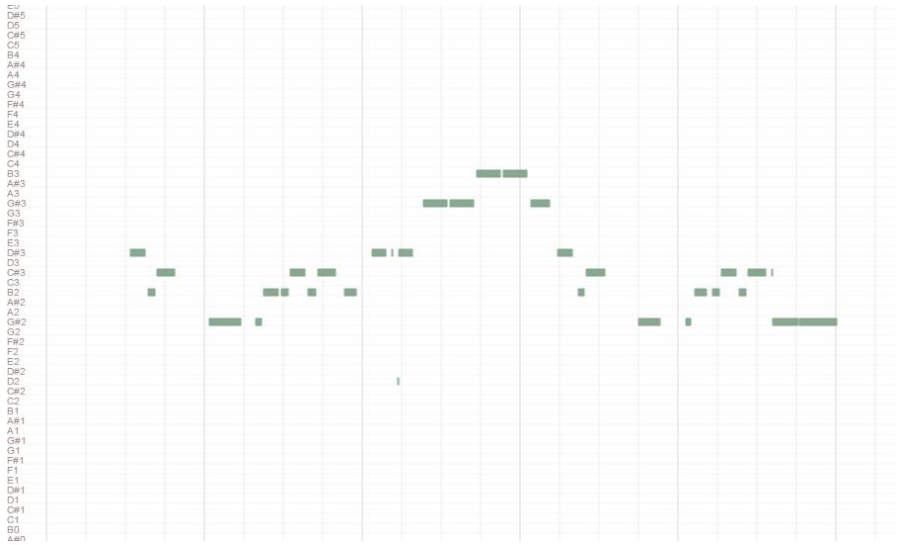
Οι δοκιμές που θα γίνουν περιλαμβάνουν πραγματικές περιπτώσεις χρήσης όπως έχουν περιγραφεί και στο εγχειρίδιο χρήσης. Πρώτα θα περιγραφούν κάποιες δοκιμές που έγιναν στο σύνολο του συστήματος ώστε να δούμε τα αποτελέσματα που καταγράφει και στη συνέχεια θα δοκιμαστούν επιμέρους μονάδες. Σαν πολύ σημαντικό κομμάτι κρίνονται οι επιδόσεις για τις οποίες και θα γίνουν μετρήσεις.

5.1.1 Σύστημα

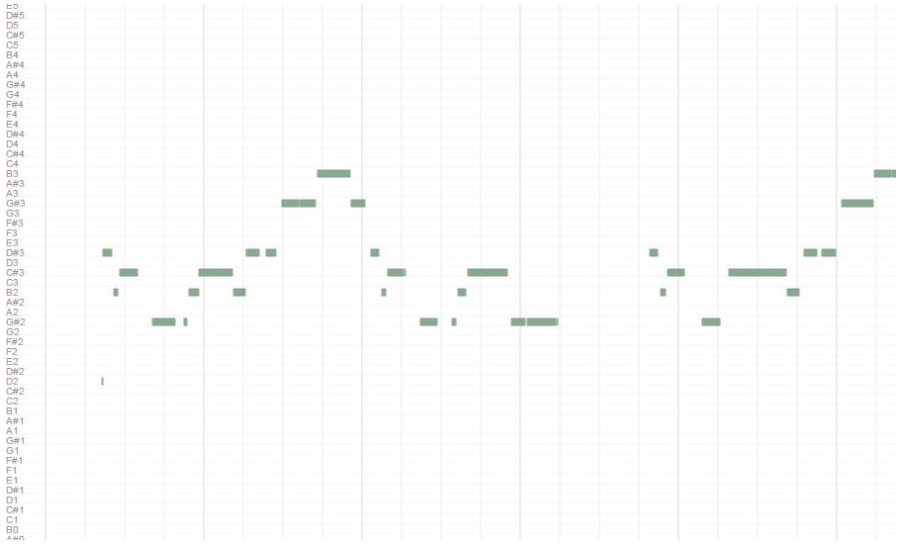
5.1.1.1 Δοκιμή στο κομμάτι “Hit the road Jack” με πιάνο και κιθάρα



5.1 Μέρος των αυθεντικών νοτών του κομματιού "Hit the road Jack"

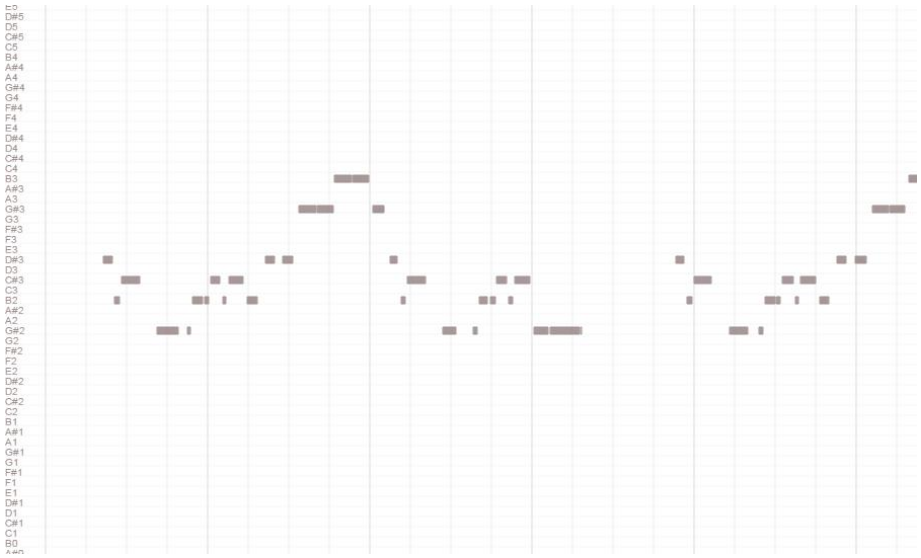


5.2 Καταγραφή μέρους του κομματιού "Hit the road Jack" παιγμένο από πιάνο στα 110 BPM, με επανασχεδιασμό πλήρους διόρθωσης

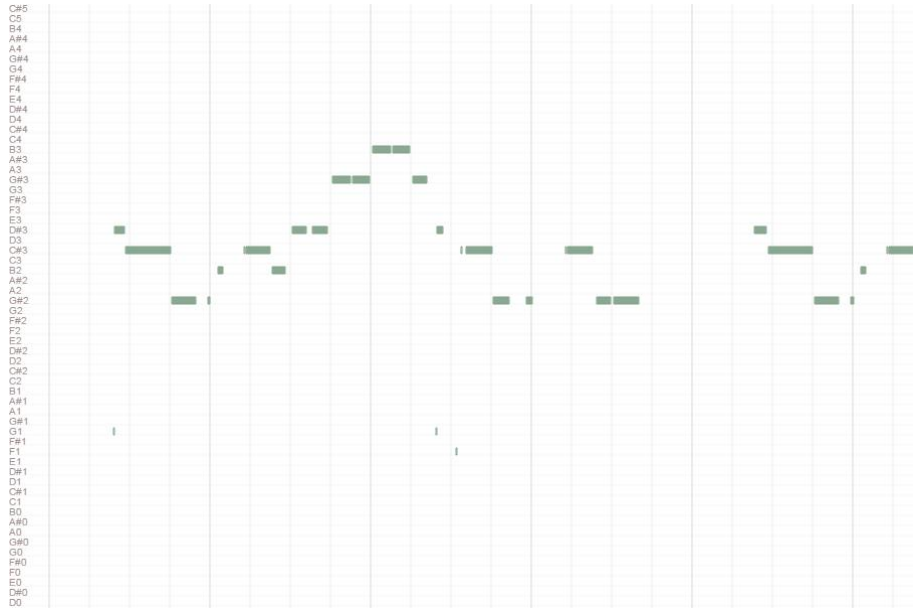


5.3 Καταγραφή μέρους του κομματιού "Hit the road Jack" παιγμένο από πιάνο στα 170 BPM, με επανασχεδιασμό πλήρους διόρθωσης

Το μουσικό όργανο είναι ηχογραφημένο υπό ιδανικές συνθήκες στούντιο χωρίς καθόλου εξωτερικό θόρυβο. Παρατηρούμε πως το αποτέλεσμα της εφαρμογής είναι αρκετά ακριβές σχετικά με το αυθεντικό. Βλέπουμε μεγαλύτερη ακρίβεια στο κομμάτι που παίζεται με ποιο αργή ταχύτητα.



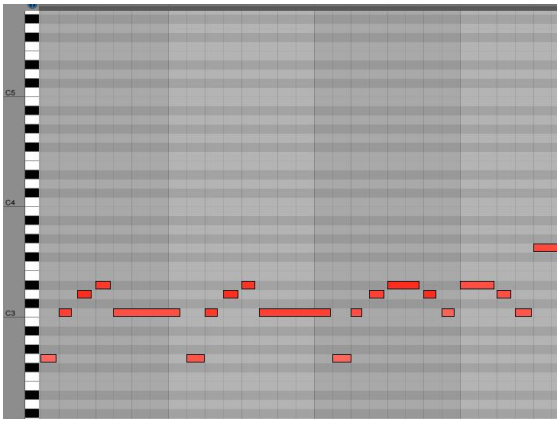
5.4 Καταγραφή μέρους του κομματιού "Hit the road Jack" παιγμένο από πιάνο στα 170 BPM, χωρίς επανασχεδιασμό πλήρους διόρθωσης



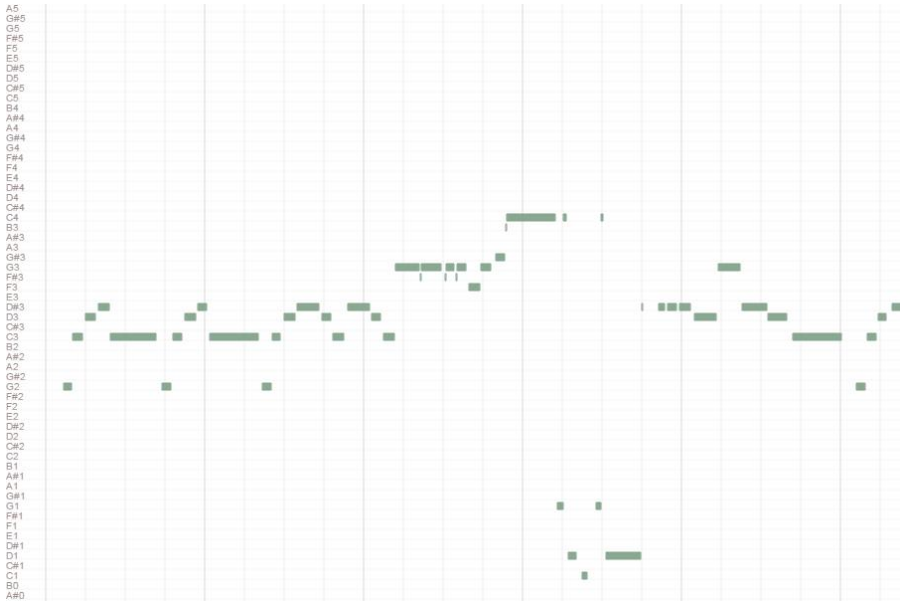
5.5 Καταγραφή μέρους του κομματιού "Hit the road Jack" παιγμένο από κιθάρα στα 150 BPM, με επανασχεδιασμό πλήρους διόρθωσης

Επίσης παρατηρούμε της αποτελεσματικότητα του επανασχεδιασμού πλήρους διόρθωσης καθώς και μία απόκλιση αποτελέσματος από όργανο σε όργανο.

5.1.1.2 Δοκιμή στο κομμάτι “Bella ciao” με τρομπέτα



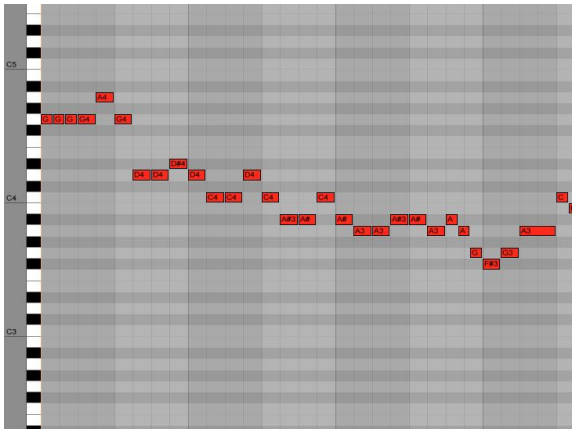
5.6 Μέρος των αυθεντικών νοτών του κομματιού "Bella ciao"



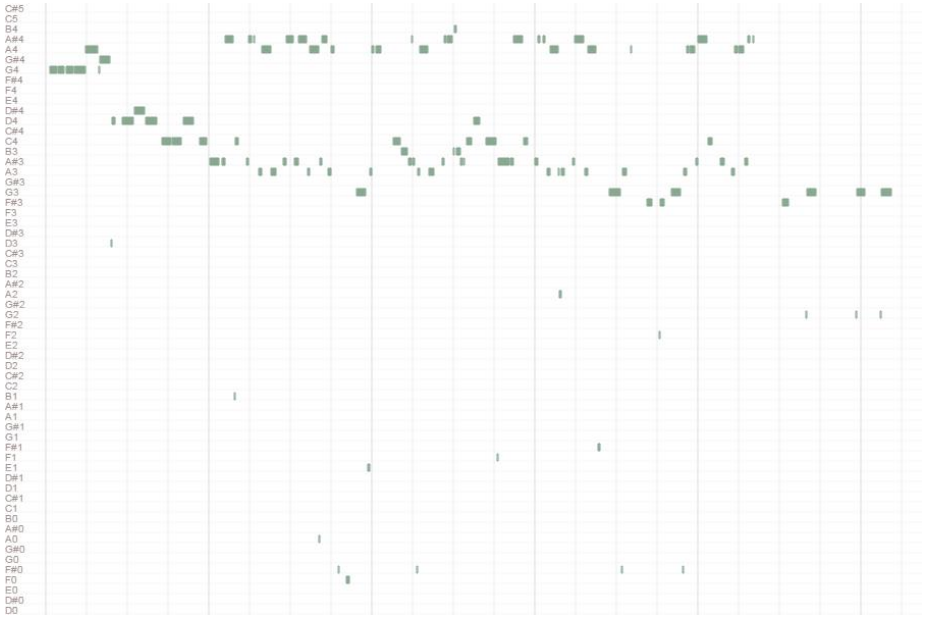
5.7 Καταγραφή μέρους του κομματιού "Bella ciao" παιγμένο από τρομπέτα στα 120 BPM, με επανασχεδιασμό πλήρους διόρθωσης

Το μουσικό όργανο είναι ηχογραφημένο υπό ιδανικές συνθήκες στούντιο χωρίς καθόλου εξωτερικό θόρυβο. Παρατηρούμε μία ακριβή καταγραφή όταν το όργανο που χρησιμοποιούμε δεν έχει αντηχείο.

5.1.1.3 Δοκιμή στο κομμάτι “Φραγκοσυριανή” με Κόρα



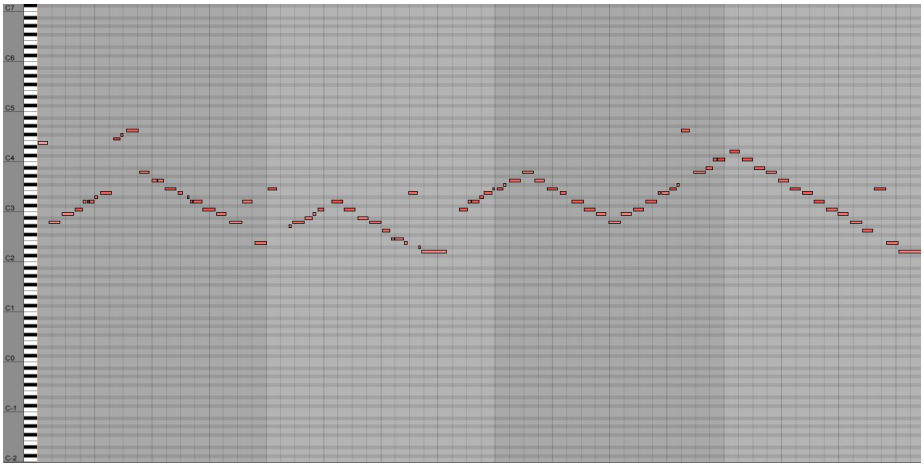
5.8 Μέρος των αυθεντικών νοτών του κομματιού "Φραγκοσυριανή"



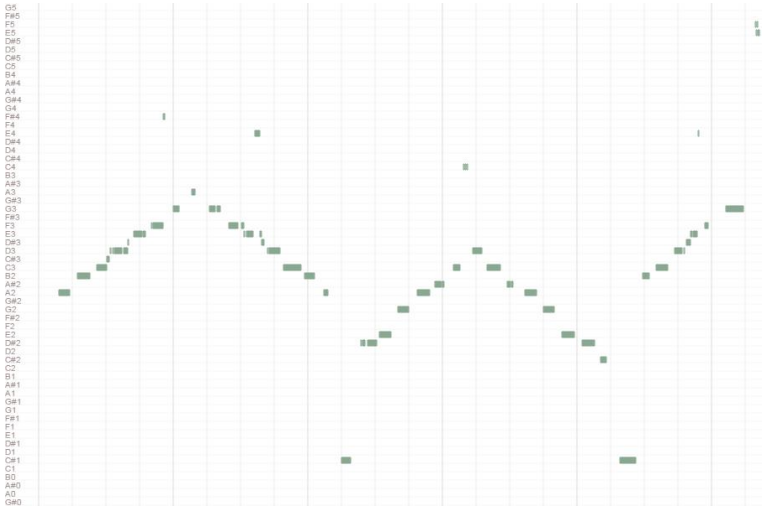
5.9 Καταγραφή μέρους του κομματιού "Φραγκοσυριανή" παιγμένο από κόρα στα 62 BPM, με επανασχεδιασμό πλήρους διόρθωσης

Το μουσικό όργανο είναι ηχογραφημένο υπό ιδανικές συνθήκες στούντιο χωρίς καθόλου εξωτερικό θόρυβο. Παρατηρούμε ότι σε μουσικά όργανα με αντηχείο μπορεί να έχουμε πληθώρα ανεπιθύμητων αποτελεσμάτων.

5.1.1.4 Σύγκριση καταγραφών με παρόμοια εφαρμογή του λογισμικού Ableton live με Μπουζούκι



5.10 Καταγραφή μουσικής πληροφορίας στο λογισμικό Ableton live από αναπαραγωγή κομματιού σε μπουζούκι

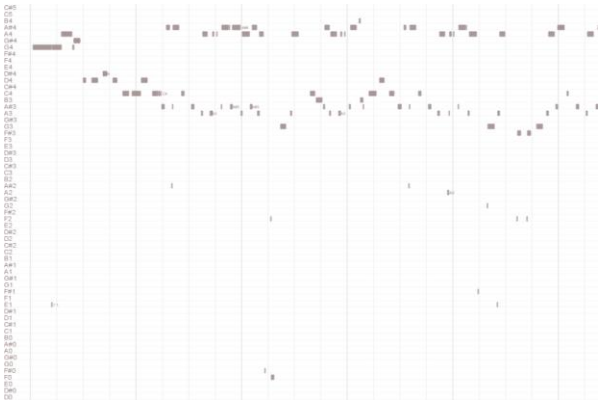


5.11 Καταγραφή μουσικής πληροφορίας στο λογισμικό Note Recognizer από αναπαραγωγή κομματιού σε μπουζούκι

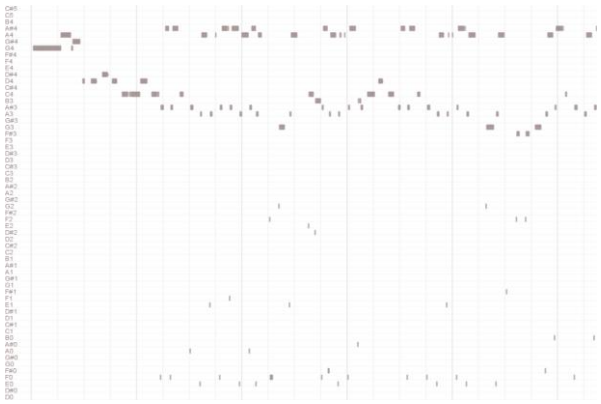
Το μουσικό όργανο είναι ηχογραφημένο με συμβατικό μικρόφωνο υπολογιστή, έτσι το σήμα έχει ένα ποσοστό θορύβου. Παρατηρούμε ότι συγκριτικά με επαγγελματικές εφαρμογές τα αποτελέσματα της Note Recognizer είναι αρκετά ικανοποιητικά.

5.1.2 Μονάδες

5.1.2.1 Δοκιμή πάνω στη λειτουργία αυτοδιόρθωσης στο κομμάτι “Φραγκοσυριανή” με Κόρα



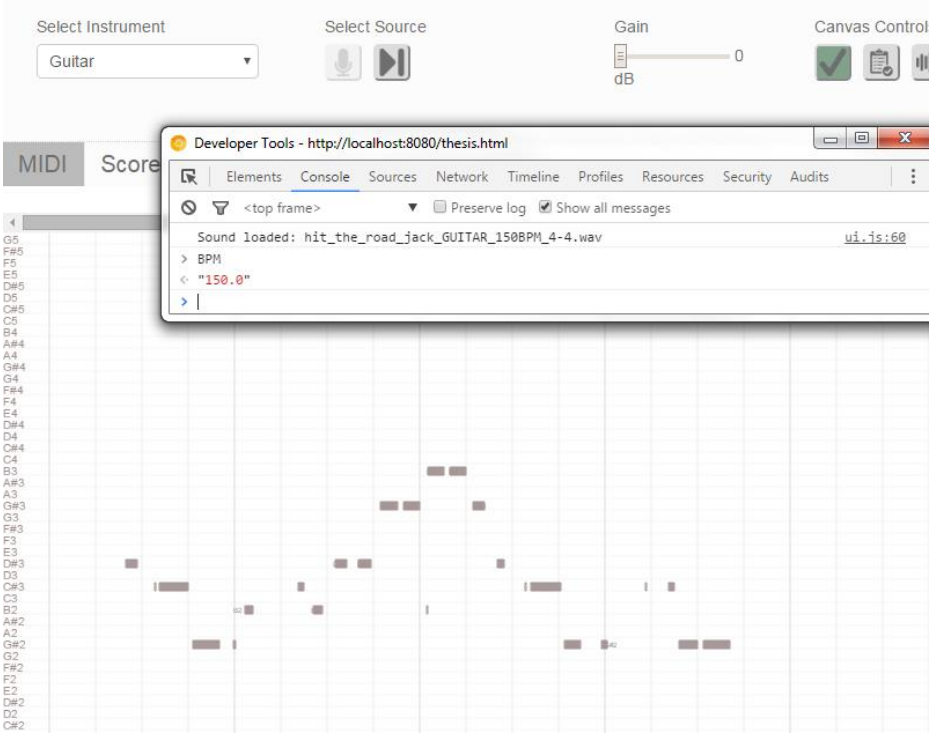
5.12 Καταγραφή μέρους του κομματιού "Φραγκοσυριανή" παιγμένο από κόρα στα 50 BPM, με την λειτουργία της αυτοδιόρθωσης ενεργοποιημένη



5.13 Καταγραφή μέρους του κομματιού "Φραγκοσυριανή" παιγμένο από κόρα στα 50 BPM, με την λειτουργία της αυτοδιόρθωσης απενεργοποιημένη

Το μουσικό όργανο είναι ηχογραφημένο υπό ιδανικές συνθήκες στούντιο χωρίς καθόλου εξωτερικό θόρυβο. Παρατηρούμε ότι η λειτουργία της αυτοδιόρθωσης παρέχει διόρθωση στο μεγαλύτερο ποσοστό των λαθών που προκύπτουν σε χαμηλότερες συχνότητες αλλά και σε ασυνέχειες κοντά ή εντός των νοτών.

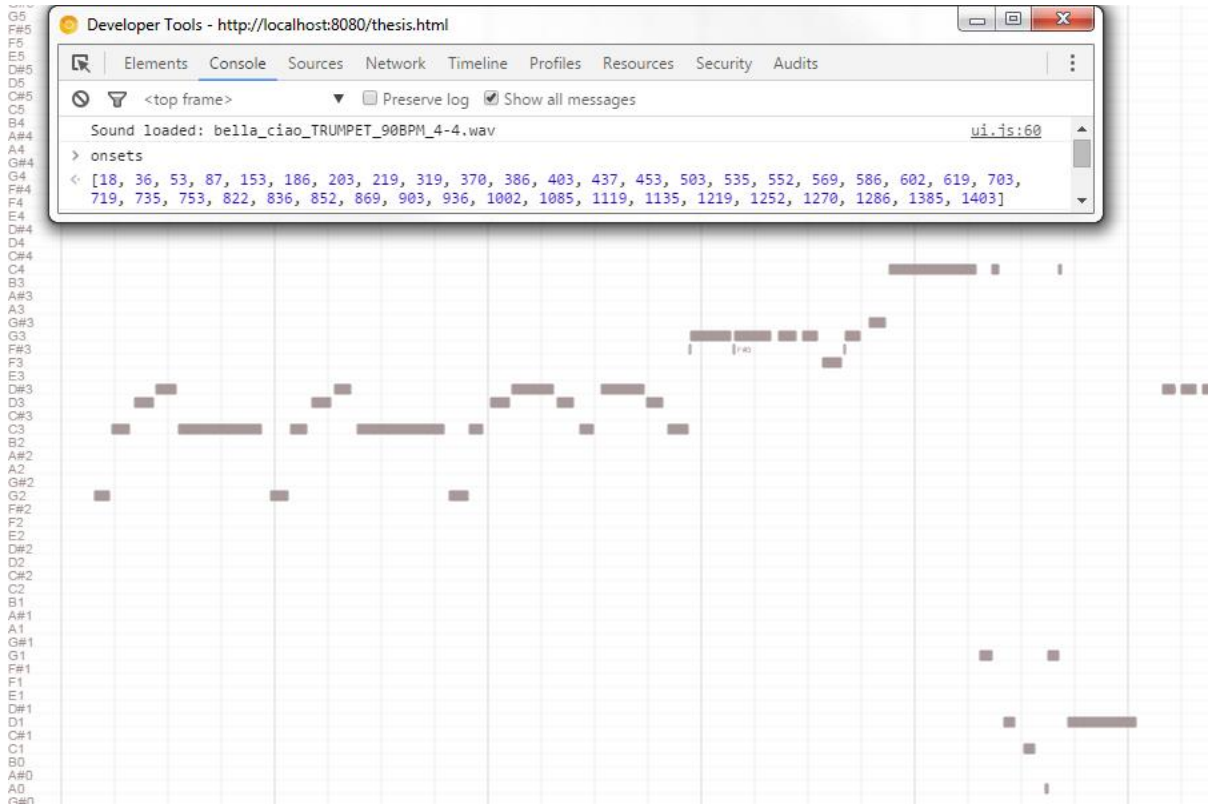
5.1.2.2 Δοκιμή πάνω στη λειτουργία υπολογισμού BPM στο κομμάτι “Hit the road Jack” με Κιθάρα



5.14 Καταγραφή μέρους του κομματιού "Hit the road Jack" παιγμένο από κιθάρα στα 150 BPM, στην κονσόλα εμφανίζεται η μεταβλητή που καταγράφεται το τελικό BPM

Το μουσικό όργανο είναι ηχογραφημένο υπό ιδανικές συνθήκες στούντιο χωρίς καθόλου εξωτερικό θόρυβο.

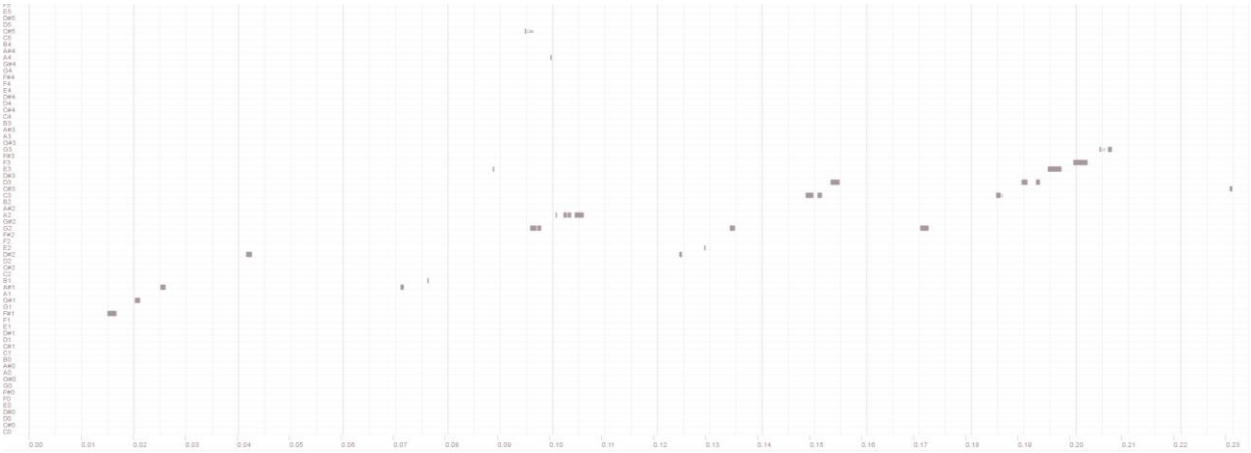
5.1.2.3 Δοκιμή πάνω στη λειτουργία εντοπισμού των onset στο κομμάτι “Bella ciao” με Τρομπέτα



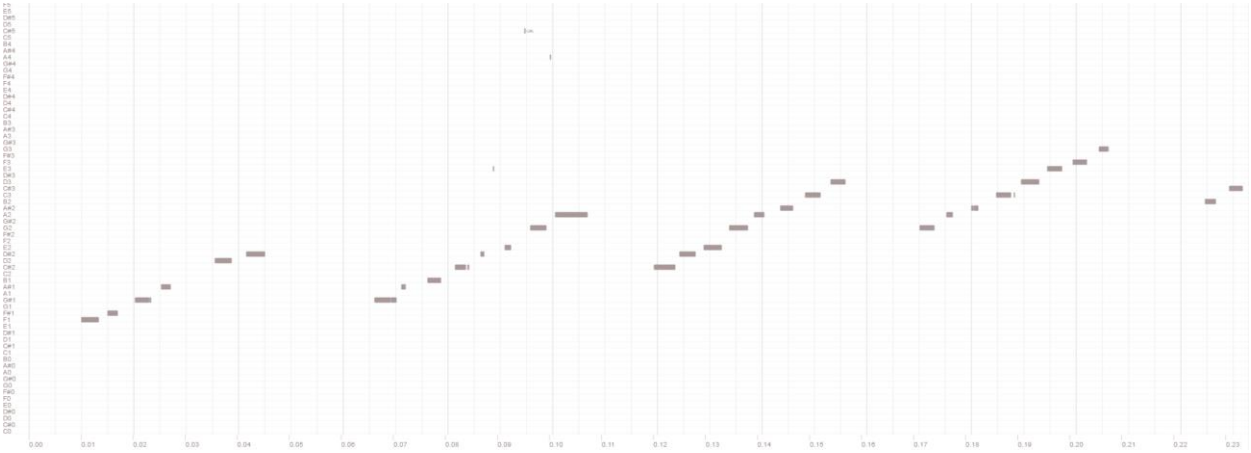
5.15 Καταγραφή μέρους του κομματιού "Bella ciao" παιγμένο από τρομπέτα στα 90 BPM, στην κονσόλα εμφανίζεται ο πίνακας όπου καταγράφονται οι θέσεις των BPM στον πίνακα στιγμιότυπων.

Το μουσικό όργανο είναι ηχογραφημένο υπό ιδανικές συνθήκες στούντιο χωρίς καθόλου εξωτερικό θόρυβο.

5.1.2.4 Δοκιμή πάνω στη λειτουργία αύξησης έντασης του εισερχόμενου σήματος με Κιθάρα



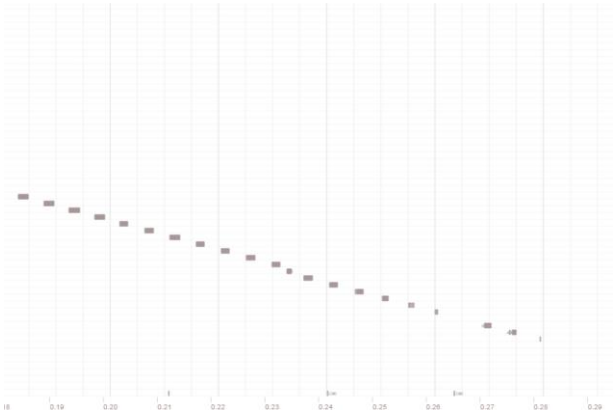
5.16 Καταγραφή χωρίς προενίσχυση σήματος



5.17 Καταγραφή με προενίσχυση σήματος

Το μουσικό όργανο είναι ηχογραφημένο με συμβατικό μικρόφωνο υπολογιστή, έτσι το σήμα έχει ένα ποσοστό θορύβου. Παρατηρούμε ότι οι αλγόριθμοι δεν μπορούν να κάνουν καταγραφή της πληροφορίας σε χαμηλής έντασης σήματα. Επίσης βλέπουμε ότι παρόλη την αύξηση έντασης ψηφιακά δεν εμφανίζεται θόρυβος, δηλαδή ασυνεχείς καταγραφές.

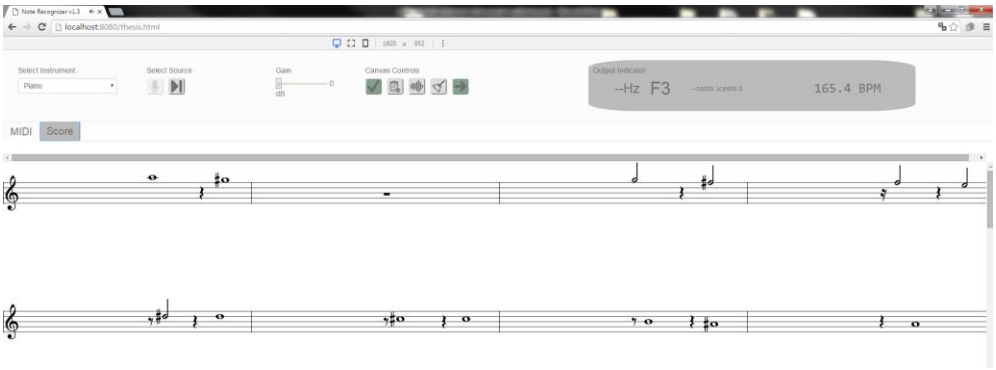
5.1.2.5 Δοκιμή πάνω στη λειτουργία εκτίμησης τονικού ύψους σε νότες χαμηλής συχνότητας με Πιάνο



5.18 Καταγραφή πιάνου - η τελευταία γραμμή του άξονα y είναι η νότα C0

Το μουσικό όργανο είναι ηχογραφημένο υπό ιδανικές συνθήκες στούντιο χωρίς καθόλου εξωτερικό θόρυβο. Παρατηρούμε ότι από ένα όριο και κάτω οι νότες δεν εντοπίζονται λόγο της χαμηλής τους συχνότητας, το οποίο συμβαίνει λόγο του μήκους του παραθύρου του AnalyserNode όπως αναλύθηκε προηγουμένως.

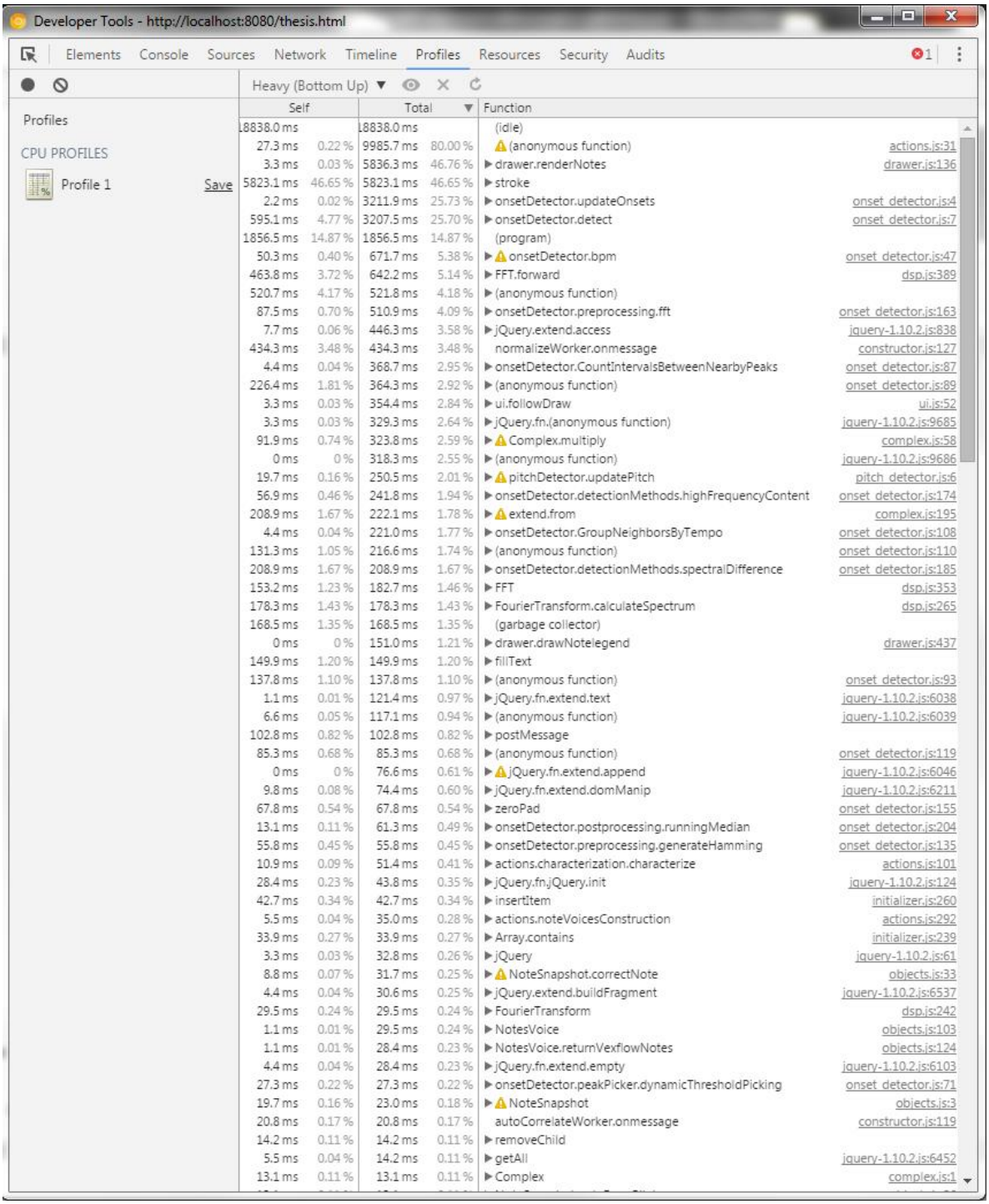
5.1.2.6 Δοκιμή πάνω στη λειτουργία καταγραφής σε παρτιτούρα με Πιάνο



5.19 Καταγραφή μουσικής πληροφορίας αποσπάσματος μουσικού κομματιου σε παρτιτούρα

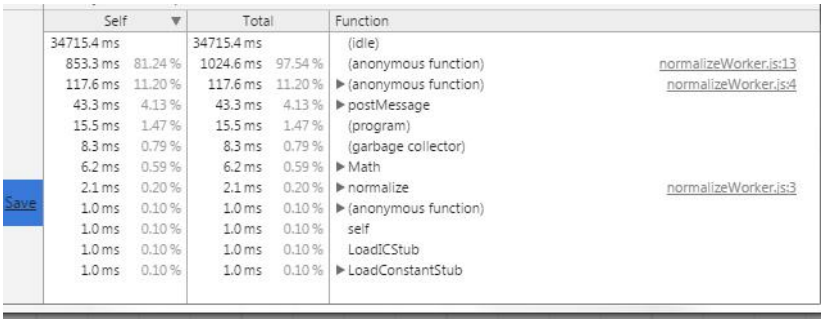
Βλέπουμε ότι η παρούσα λειτουργία είναι τελείως πειραματική.

5.1.3 Επιδώσεις

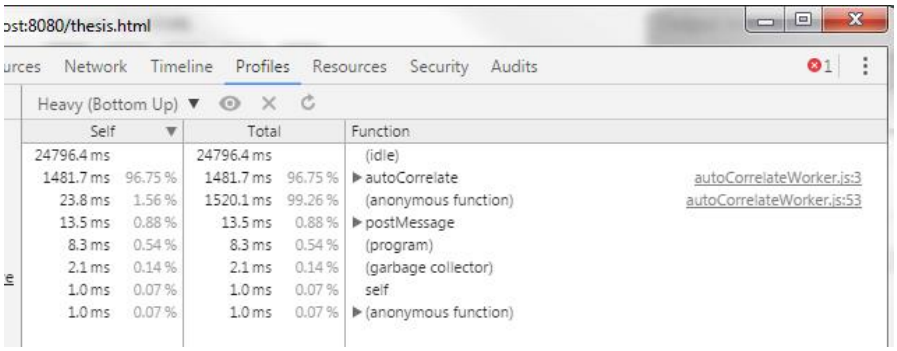


5.20 Λίστα που δείχνει που καταναλώνεται ο χρόνος εκτέλεσης μεταξύ των συναρτήσεων του κορμού της εφαρμογής

Παρατηρούμε ότι η εφαρμογή έχει πολύ υψηλές υπολογιστικές απαιτήσεις. Οι πιο δαπανηρές συναρτήσεις της εφαρμογής βλέπουμε ότι είναι ο σχεδιασμός και ο εντοπισμός των onset. Η απαιτήσεις φυσικά δεν θα ήταν οι ίδιες αν δεν χρησιμοποιούνταν νήμα εκτέλεσης ή αυτό που χρησιμοποιείται εκτελούνταν λιγότερες φορές το δευτερόλεπτο. Οι δοκιμές έγιναν σε υπολογιστή με επεξεργαστή Intel Core i5-4460 3.20 GHz, μνήμη RAM 8 GB και λειτουργικό σύστημα Windows 7 Ultimate 64-bit.



5.21 Λίστα που δείχνει που καταναλώνεται ο χρόνος εκτέλεσης μεταξύ των συναρτήσεων του worker κανονικοποίησης



5.22 Λίστα που δείχνει που καταναλώνεται ο χρόνος εκτέλεσης μεταξύ των συναρτήσεων του worker αυτοσυσχέτησης

5.2 Συμπεράσματα

Από τη συγκεκριμένη πτυχιακή εργασία ωφελείτε ο σπουδαστής σε μεγάλο βαθμό λόγο της έρευνας και κατανόησης του συγκεκριμένου θέματος, καθώς και λόγο των αυξημένων γνώσεων που απέκτησε σε τεχνολογικό επίπεδο. Επίσης η εφαρμογή ως μονάδα μπορεί να χρησιμοποιηθεί από μουσικούς για να έχουν μία γεύση του τι παίζουν τη στιγμή που το παίζουν, το οποίο μπορεί να πει κανείς ότι μπορεί να τους ωφελήσει εκπαιδευτικά προς το παρόν ή να τους βοηθήσει στην απομνημόνευση πρωτότυπων κομματιών.

Αναφορικά με τις περιγραφές που ορίστηκαν στο κεφάλαιο 4.2 βλέπουμε ότι η εφαρμογή παρουσιάζει εκ του αποτελέσματος επίτευξη των βέλτιστων προδιαγραφών για το μεγαλύτερο μέρος των λειτουργιών της.

5.3 Μελλοντική εργασία και επεκτάσεις

Η παρούσα έκδοση της εφαρμογής υλοποιεί ένα μέρος των αλγορίθμων που έχουν μελετηθεί. Αποφασίστηκε η παρουσίαση περιορισμένου εύρους λειτουργικοτήτων ώστε η εφαρμογή να είναι κατ 'αρχάς σταθερή, λειτουργική και να δίνει το στίγμα των μελλοντικών επιδιώξεων της. Σημαντικό κομμάτι στη μελλοντική εργασία είναι η σύγκριση τεχνικών που δημιουργήθηκαν κατά την διαδικασία υλοποίησης της εφαρμογής από το φοιτητή με ανάλογες τεχνικές αναλύσεις που προϋπάρχουν και είναι επιστημονικά αποδεδειγμένες, ώστε να διαπιστωθούν ελαττώματα και βελτιωμένες λύσεις.

5.3.1 Λειτουργίες της εφαρμογής προς βελτίωση και επέκταση

- Διαχωρισμός διεργασιών με υψηλές υπολογιστικές απαιτήσεις σε επιπλέον workers.
- Απεικόνιση της μουσικής σημειογραφίας σε παρτιτούρα
- Περαιτέρω ανάπτυξη της λειτουργίας αυτοδιόρθωσης
- Επιλογή βέλτιστων ρυθμίσεων ανάλογα με την επιλογή τύπου οργάνου όχι μόνο στην αποκοπή περιττών συχνοτήτων αλλά και στην επιλογή των κατάλληλων αλγορίθμων για την καταγραφή

- Βελτίωση της δομής του κώδικα διαχωρίζοντας τον σε επιπλέον κλάσεις οι οποίες θα είναι περισσότερο αυτόνομες.
- Αντικατάσταση του αλγορίθμου εύρεσης των offset με κάποιον επιστημονικά αποδεδειγμένο.
- Αντικατάσταση AnalyserNodes με AudioWorkers όταν υπάρξει συμβατότητα των τελευταίων με τους φυλλομετρητές.
- Βελτίωση του τρόπου που εναλλάσσετε η παρουσίαση της πληροφορίας κατά την αλλαγή ταχύτητας.
- Αυτόματη επέκταση canva όταν τελειώνει ο χώρος καταγραφής

5.3.2 Νέες λειτουργίες προς μελλοντική υλοποίηση

- **Midi Editor, Canvas tools, MIDI export και αναπαραγωγή καταγραφής**

  Δυνατότητα στο χρήστη να επεξεργάζεται τα καταγεγραμμένα αποτελέσματα με γραφικό τρόπο και οι αλλαγές να διαδρούν με τα αντικείμενα του συστήματος. Επίσης εργαλεία όπως η αύξηση του μεγέθους του canva και κέρσορες για την κατάδειξη της θέσης στους άξονες με την ανάλογη πληροφορία. Επίσης παροχή δυνατότητας στο χρήστη να εξάγει τη μουσική πληροφορία σε αρχείο mid καθώς και να μπορεί να αναπαράγει το αποτέλεσμα της καταγραφής.

- **Εισαγωγή αυξημένων επιλογών και απεικόνισης στη διεπαφή χρήστη**

  Δυνατότητα του χρήστη να κάνει επιλογές για προχωρημένους ώστε να βελτιστοποιήσει το αποτέλεσμα καθώς και απεικόνισης του σήματος σε δείκτη έντασης και φασματογράφο.

- **Αύξηση συχνότητας εκτέλεσης του νήματος εκτέλεσης**

  Ανάπτυξη τεχνικών που θα βοηθήσουν στην αύξηση της συχνότητας εκτέλεσης του βασικού thread ώστε να υπάρχει δυνατότητα βελτιωμένης καταγραφής και σε συνθήκες γρήγορης αναπαραγωγής αλλά και η δυνατότητα αναγνώρισης νοτών μικρής διάρκειας.

- **Ανάπτυξη αλγορίθμων ή χρήση υπαρχόντων για την αποκοπή θορύβου που προέρχεται από την αντήχηση του οργάνου.**

- **Εισαγωγή πολύπλοκων μουσικών αξιών προς αναγνώριση όπως παρεστιγμένα, τρίηχα κλπ.**
- **Καταγραφή των αποτελεσμάτων σε ταμπλαδούρα**

  Καταγραφή σε ταμπλαδούρα αλλά και απεικόνιση της τρέχουσας νότας σε μοντέλο μουσικού οργάνου.

- **Εύρεση time signature, ρυθμού και δρόμου της εκτέλεσης**

  Ανάπτυξη συστήματος μάθησης για την περαιτέρω ανάλυση της μουσικής πληροφορίας ανάμεσα σε υπάρχοντα μουσικά μοντέλα.

- **Αυτόματος εντοπισμός τύπου οργάνου από το ηχόχρωμα**
- **Ανάπτυξη αλγορίθμων για την καταγραφή πολυφωνικής μουσικής πληροφορίας**

  Ανάπτυξη αλγορίθμων για καταγραφή και συγχορδιών.

- **Ανάπτυξη περιβάλλοντος απομακρυσμένης μουσικής σύμπραξης με βάση την παρούσα εφαρμογή και τις δυνατότητες της**

  Δυνατότητα σε δύο απομακρυσμένους χρήστες να ανταλλάξουν την πληροφορία που το σύστημα καταγραφεί σε πραγματικό χρόνο.

Βιβλιογραφία

[1] Ζαχαράτου, Ε. Τ. (2013). Αυτόματη Καταγραφή Μουσικής. Αθήνα: Εθνικό Μετσόβιο Πολυτεχνείο.

[2] Mediacollege. (n.d.). Retrieved from http://www.mediacollege.com/audio/01/sound-waves.html

[3] Σγουρός, Ν. Ηχητικά συστήματα. http://users.iit.demokritos.gr/~ntsap/courses/bes04/various/sgouros02.pdf.

[4] Musicjourneys. (n.d.). Retrieved from https://musicjourneys.wordpress.com/2013/03/10/6/

[5] Maya-gaia. (n.d.). Retrieved from http://maya-gaia.angelfire.com/primordial_rhythm_meditate.html

[6] Smus, B. Web Audio API. O'Reilly.

[7] Wikipedia. (n.d.). Retrieved from https://en.wikipedia.org/wiki/HTML5

[8] Mozillademos. (n.d.). Retrieved from https://mdn.mozillademos.org/files/8857/web-audio-api-flowchart.png

[9] Mozillademos. (n.d.). Retrieved from https://mdn.mozillademos.org/files/8859/WebAudioGenerics.png

[10] Mozilla. (n.d.). Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Basic_concepts_behind_Web_Audio_API

[11] Mozilla. (n.d.). Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/AnalyserNode

[12] Mozilla. (n.d.). Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/ScriptProcessorNode

[13] Github. (n.d.). Retrieved from http://webaudio.github.io/web-audio-api/#idl-def-AudioWorkerNode

[14] Github (n.d.). github. Retrieved from https://webaudio.github.io/web-audio-api/#the-dynamicscompressornode-interface

[15] Wikipedia. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Filter_%28signal_processing%29

[16] Mozilla. (n.d.). Retrieved from https://developer.mozilla.org/el/docs/Web/API/BiquadFilterNode

[17] Wikimedia. (n.d.). Retrieved from https://upload.wikimedia.org/wikipedia/en/thumb/e/ec/Bandform_template.svg/400px-Bandform_template.svg.png

[18] Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., & Sandler, M. B. (2005). A Tutorial on Onset Detection in Music Signals. IEEE.

[19] Blogspot. (n.d.). Retrieved from http://tabloleung.blogspot.gr/2015/05/6-onset-detection.html

[20] Wordpress. (n.d.). Retrieved from https://eecs4life.wordpress.com/concepts/

[21] Det, J. (n.d.). slideshare. Retrieved from http://www.slideshare.net/janessadet/

[22] Rosão, C., Ribeiro, R., & Matos, D. M. Comparing Onset Detection Methods Based on Spectral.

[23] Man, T. K. (2006). Tempo Extraction using the Di. Hong Kong: Hong Kong University of Science and Technology .

[24] Cai, W. (2013). Analysis of Acoustic Feature Extraction. New York: University of Rochester.

[25] Independentrecording. (n.d.). Retrieved from http://www.independentrecording.net/irn/resources/freqchart/images/main_chart.jpg

[26] Wikipedia. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Autocorrelation

[27] Cnx. (n.d.). Retrieved from http://cnx.org/contents/8b900091-908f-42ad-b93d-806415434b46@2/Pitch-Detection-Algorithms